On the Security of Multi-Party Ping-Pong Protocols (fragments of a revised version dating to July 1985)

Shimon Even[1]
Department of Computer Science
Technion - Israel Institute of Technology
Haifa, ISRAEL.
E-mail: `even@cs.technion.ac.il`

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, ISRAEL.
E-mail: `oded@wisdom.weizmann.ac.il`

Research done mainly in Summer 1982[2]
Original version appeared as a TR[3] in June 1983
Revised version written in July 1985[4]
Abstract and Introduction reproduced in February 1996
Technical part reproduced and slightly revised in February 2004[5]

[1]Partially supported by the Fund for the Promotion of Research at the Technion.

[2]While the authors visited the EECS Dept., UC-Berkeley, and were supported by NSF grant No. MCS 82-04506.

[3]TR No. 285, Computer Science Dept., Technion, Haifa, Israel, June 1983. Holding 59 pages.

[4]At that time the second author was at MIT, where he was partially supported by a Weizmann Post-Doctoral Fellowship.

[5]At that time the second author was at Radcliffe Institute for Advanced Study at Harvard University, where he was partially supported by a Radcliffe Fellowship.

Preface by Oded (2004)

This report was reproduced from files dating to 1985. These files contained only a small fragment of a working draft of an intended revision of the original write-up.[1] The original (59-page) write-up (written in 1982/83) has appeared as a Technical Report (No. 285 of the Computer Science Dept., Technion, Haifa, Israel, June 1983). An extended abstract has appeared in the *24th FOCS*, Nov. 1983.

The orignal work refers to a restricted notion of insecurity (i.e., breakability under a syntactically restricted type of attacks) and to restricted classes of protocols. These classes extend the model of Dolev and Yao in two ways. First, the work considers multi-party protocols rather than two-party ones. Next, the work considers protocols in which each message consists of a pair of strings and possibly different operations are applied to each element in the pair (rather than allowing only operations that are applied to the message as a whole). The focus of the work is on the complexity (or even decidability) of the computational task of testing whether or not such protocols are insecurity (under the aforementioned type of attacks). The results include:

- For every fixed m, a polynomial-time algorithm for the testing problem of m-party protocols of the Dolev-Yao type.
- The testing problem for multi-party protocols of the Dolev-Yao type is NP-Hard, where m is part of the input (rather than being fixed).
- The testing problem for two-party protocols that operate on pairs of strings is undecidable.

The incentive to reproduce these files almost two decades after they were written is to provide a more accessible account of the original work, in light of the renewed interest in it. In addition to converting these files from `troof` to LaTeX, I have mildly proofread the text and modified it a little (without modifying the overall structure or the low-level style). In a few places I have added footnotes and "new notes" (where the latter are explicitly marked as such).

[1]The revision was intended to be quite drastic. Specifically, the entire paper was to be written from scratch, while using the original work [EG] only as a source of ideas. Indeed, the existing working draft deviates significantly (especially in the technical part) from the original presentation (although no significant new idea was introduced).

Contents

Abstract

This paper is concerned with the model for security of cryptographic protocols suggested by Dolev and Yao. The Dolev and Yao model deals with a restricted class of protocols, known as *Two-Party Ping-Pong Protocols.* In such a protocol, messages are exchanged in a memoryless manner. That is, the message sent by each party results from applying a predetermined operator to the message he has received.

The Dolev and Yao model is presented, generalized in various directions and the affect of these generalizations is extensively studied. First, the model is trivially generalized to deal with multi-party ping-pong protocols. However, the problems which arise from this generalization are very far from being trivial. In particular, it is no longer clear how many saboteurs (adversaries) should be considered when testing the security of p-party ping-pong protocols. We demonstrate an upper bound of $3(p-2)+2$ and a lower bound of $3(p-2)+1$ on this number. Thus, for every fixed p, the security of p-party ping-pong protocols can be tested in polynomial time. In contrast, we show that testing the security of multi-party protocols (i.e. the number of participants is part of the input) is NP-Hard. A different extension of the Dolev and Yao model, obtained by allowing operators to operate on "half words", is shown to have an undecidable security problem.

Keywords: Cryptographic Protocols, Security, Public-Key Cryptosystems, String Replacement Problems, Undecidability, Concrete Complexity, NP-Completeness, Combinatorial Analysis, Routing Problems, Graph Theory.

Chapter 1

Introduction

The use of public-key encryption [DH, RSA] for secure network communication has received considerable attention. Such systems are effective against a "passive" eavesdropper, namely one who merely taps the communication line and tries to decipher the intercepted messages. However, as pointed out by Needham and Schroeder [NS], an improperly designed protocol can be vulnerable to "active" sabotage.

The "active" saboteur (adversary) may be a legitimate user in the network. He can intercept and alter messages, impersonate other users, or initiate instances of the protocol between himself and other users in order to use their responses. It is possible that through such complex manipulations he can read messages that are supposed to be protected without cracking the cryptosystem in use.

In view of this danger it is desirable to have a formal model for discussing security issues in a precise manner. The first such model was introduced by Dolev and Yao [DY], and constitutes the subject of this paper. The Dolev and Yao model consists of a restricted class of "memoryless protocols" and a related definition of insecurity. Loosely speaking, a protocol is insecure if there is a way to obtain the initial message (which is transferred by it), even if the public-key encryption in use are "ideal". This insecurity definition captures all possible weaknesses in the "high level structure" of the protocol; that is, weaknesses that are independent of the particular encryption function used to implement the abstract protocol.

Dolev and Yao considered two-party protocols that proceed in phases as follows. In the first phase the "first" party applies a predetermined sequence of encryption and decryption operators to an *initial message* of his choice and transmit the result. In each later phase, a predetermined party applies a predetermined operator sequence to the last message he/she received and transmits the result. The set of operators was later extended to contain name appending/deletion operators, and the resulting protocols were called *ping-pong* protocols. The related insecurity definition captures all possible "generic" manipulations that the saboteurs can apply to messages they intercept, by possibly using "replays" of the very protocol. In "generic manipulations" we mean actions which do not depend on the specific cryptosystem in use, but rather relate only to the "high level structure" of the protocol. (More details are given in section 2.1.)

Dolev and Yao have demonstrated that testing the security of a two-party ping-pong protocol can be done in polynomial time. A much more efficient algorithm was presented by Dolev, Even and Karp [DEK]. Its running time is $O(n^3)$, where n is the length of the input. The purpose of this paper is to further investigate the Dolev and Yao model by considering two natural extensions of it.

1. First we consider *multi*-party ping-pong protocols. This naive-looking extension causes a lot of trouble. In contrast to the case of two-party ping-pong protocols, where it was sufficient to

consider the actions of a single saboteur, the situation in the general case is more involved: At least $3(p-2)+1$ saboteurs must be considered for testing the security of a p-party ping-pong protocols. On the other hand, we show that $3(p-2)+2$ saboteurs suffice for this purpose. Using this upper bound, a natural extension of [DEK] implies that, for every fixed p, there is a polynomial-time algorithm for testing the security of p-party ping-pong protocols. For unfixed p this is not likely to be the case, since we show that testing the security of multi-party ping-pong protocols is NP-Hard (here p the number of participants is part of the input).

2. Next, we slightly extend the simple "operator vocabulary" by introducing operators that operate on "half words" (i.e., messages are viewed as pairs of strings, and operators are allowed to operate on one element of the pair while leaving the other element intact).[1] It is shown that testing the security of protocols in this class is non-recursive.

Organization of the Paper

The rest of the paper is partitioned to three parts. The first part (Chapter 2) deals with multi-party ping-pong protocols, the second (Chapter 3) with the "half word" operators, and the third part (Chapter 4) contains various comments and conclusions.

[1]The original text read *Next, we slightly relax the "memoryless condition" by introducing operators that operate on "half words".* Operation on one element of the pair was viewed as slightly violating the memoryless condition because the other element (left intact) is meanwhile stored in memory. In retrospect we prefer the pharsing put in the main text.

Chapter 2

The Multi-Party Extension

In this chapter we investigate the affect of extending the Dolev and Yao model to protocols for more than two parties. We begin this chapter by presenting a formal definition of multi-party ping-pong protocols and the related security problem (section 2.1). The core of this chapter consists of lower and upper bounds on the number of saboteurs that should be considered in testing the security of p-party ping-pong protocols (section 2.2). The upper bound plays a key role in obtaining a polynomial-time algorithm for testing the security of p-party ping-pong protocols, for any fixed p. We continue by proving that testing the security of multi-party ping-pong protocols is NP-Hard, when the number of parties is part of the input (section 2.3). Section 2.4 deals with several variations of the insecurity definition.

2.1 Security of Ping-Pong Protocols: Definitions and Terminology

Let us first give a short outline of the definitions presented in this section. A multi-party ping-pong protocols proceeds in steps as follows. In the first step the "first" party applies a predetermined sequence of operators to an *initial message* of his choice and transmit the result. In each later step, a predetermined party applies a predetermined operator sequence to the last message he/she received and transmits the result. A formal definition requires a specification of the operators as well as the relations between them (see subsection 2.1.1). Once this is done, protocols and instances/executions of them are formally defined (subsection 2.1.2); and a formal definition of insecurity is presented (subsection 2.1.3). This definition is further extended in subsection 2.1.4. We end this section by presenting terminology and simple observations concerning insecure protocols (subsection 2.1.5).

2.1.1 Operators and Relations between them

We begin this subsection by an informal discussion of the operators we would like to consider. The first and most important type of operators we consider are Public-Key Encryption and Decryption. These operators are the origin of the possible security of the protocols we will consider. Following [DH], a *public key cryptosystem (PKCS)* is a set of pairs of operators, such that every user u has an encryption operator E_u and a decryption operator D_u . Both operators are mappings from $\{0,1\}^*$ to $\{0,1\}^*$. There is a public directory containing all (u, E_u) pairs, while the decryption operator D_u is known only to user u. It is required that

(1) For every $m \in \{0,1\}^*$, it holds that $E_u(D_u(m)) = D_u(E_u(m)) = m$.
(D_u is the inverse operator of E_u.)[1]

(2) On input $m \in \{0,1\}^*$, any user can efficiently compute $E_u(m)$ but only user u can efficiently compute $D_u(m)$.

For further details consult [DH] and [RSA]. More generally, we may assume that each user is associated several different instances of the Public-Key Cryptosystem; that is, several pairs of encryption decryption operators. In such a case we will denote the instances associated to user u by $(E_u^{(1)}, D_u^{(1)})$, $(E_u^{(2)}, D_u^{(2)})$, etc.

A different type of operators we consider are name appending/deletion [NS]. Applying a_u to the message m, appends u's name to it; while applying d_u to the result (i.e., $a_u(m)$), yields the original message m. The result of applying d_u to an arbitrary message may not be defined. Every user (even not u) can apply a_u and d_u . Again, there may be several distinct name appending/deletion operators per user.

Another type of operators we consider are permutations (of the message space) that are associated to certain users, although every other user can apply them. For example, each user may select randomly a key to a cryptosystem and make this key public. This by itself will not provide him with any security, nevertheless the scrambling properties of the cryptosystem still hold and may be of use for various purposes.

Operators, user's vocabulary and cancellation rules. We now turn to a formal definition of the operators. We will consider operators over the set of all bit strings. In general, each operator f_i consists of a *form*(f) and an *index*(i). For example, the operator E_u (the encryption operator of party u) is of the form E and has index u. The set of operators is symmetric in the sense that if it contains the operator f_u for some user u then it contains also the operator f_v (for every user v). In other words, the set of operators consists of all possible pairs of forms and indices. Formally

Definition 1 (operators): *Let F be a finite set and I be an arbitrary set.*[2] *Then, the* set of operators *with* forms *F and* indices *I is denoted by $\Sigma = \Sigma(F, I)$ and is defined as $\{f_i : f \in F , i \in I\}$.*

The set of forms will be partitioned to six categories: *Encryptions, Decryptions, appendings, deletions, functions* and *function-inverses*, denoted F_E, F_D, F_{app}, F_{delt}, F_{fun} and F_{inv} respectively. The reader may consider the simple case in which $F_E = \{E\}$, $F_D = \{D\}$, $F_{app} = \{a\}$, $F_{delt} = \{d\}$ and both F_{fun} and F_{inv} are empty (i.e. each user is associated a single public-key instance and a single name appending/deletion mechanism). This case will suffice for our lower bound (subsection 2.2.3) and NP-Hardness (section 2.3) results. The upper bound (subsection 2.2.4) holds also for the general case. In general, it is postulated that $|F_E| = |F_D|$, $|F_{app}| = |F_{delt}|$ and $|F_{fun}| = |F_{inv}|$. These categories correspond to the different types of operators discussed in the motivation (above). The formal role of these categories will become clear throughout the following definitions.

Definition 2 (user's vocabulary): *The* vocabulary *of user u, denoted Σ_u, consists of $\{f_i : f \in F \setminus F_D , i \in I\} \cup \{f_u : f \in F_D\}$.*

[1]The case in which D_u is only a left (right) inverse of E_u is dealt in subsection 2.4.1.

[2]We do not assume that I is finite. In fact, we will focus on the case that I is countable (e.g., isomorphic to the natural numbers). The point is that we wish to study the number of users that should be considered regarding the insecurity of p-party protocols, and the former number may depend arbitrarily on the parameter p.

That is, the vocabulary of a user contains all operators except for the decryption operators of other users. The vocabulary of a user will represent the set of operators that the user can apply to arbitrary messages.

Definition 3 (cancellation rules): *Let $F_E = \{E^{(1)}, E^{(2)}, ..., E^{(n_1)}\}$, $F_D = \{D^{(1)}, D^{(2)}, ..., D^{(n_1)}\}$, $F_{app} = \{a^{(1)}, a^{(2)}, ..., a^{(n_2)}\}$, $F_{delt} = \{d^{(1)}, d^{(2)}, ..., d^{(n_2)}\}$, $F_{fun} = \{g^{(1)}, g^{(2)}, ..., g^{(n_3)}\}$ and $F_{inv} = \{h^{(1)}, h^{(2)}, ..., h^{(n_3)}\}$. The* operator cancellation rules *consists of the following ordered pairs:*

- $(D_i^{(j)}, E_i^{(j)})$ *and* $(E_i^{(j)}, D_i^{(j)})$, *for every* $i \in I$ *and* $1 \leq j \leq n_1$.
- $(d_i^{(j)}, a_i^{(j)})$ *and for every* $i \in I$ *and* $1 \leq j \leq n_2$.
- $(h_i^{(j)}, g_i^{(j)})$ *and* $(g_i^{(j)}, h_i^{(j)})$, *for every* $i \in I$ *and* $1 \leq j \leq n_3$.

The meaning of the pair (σ, τ) is that applying the operator τ to a message m, and then applying the operator σ to the result, yields the original message m (i.e. $\sigma(\tau(m)) = m$ for every m). Note that pairs of forms are put in the suitable categories according to their role in the user's vocabularies (i.e., (F_E, F_D) versus the others) and according to the cancellation rules that refer to them (i.e., (F_{app}, F_{delt}) versus the others). A more detailed discussion of this partition can be found in subsection 2.4.1.

The Algebra of Operator Sequences. We now consider Σ^*, the set of words over $\Sigma = \Sigma(F, I)$. A word in Σ^* has a natural interpretation as sequential application of operators: the empty word (denoted by λ) corresponds to the identity operator; and the word $\alpha_2\alpha_1$ corresponds to first applying α_1 and then applying α_2 to the result. The cancellation rules induce a natural term algebra on the set Σ^*, when one considers the equivalence classes imposed by them. An alternative and more elaborate definition follows.

Definition 4 (equivalence of operator sequences): *The relation $\equiv$ is an equivalence relation defined recursively over Σ as follows.*

1. Application of a cancellation rule: *For every $\alpha, \beta \in \Sigma^*$ and a cancellation rule (σ, τ), it holds that $\alpha\sigma\tau\beta \equiv \alpha\beta$.*
2. Reflexivity, symmetry and transitivity: *For every $\alpha, \beta, \gamma \in \Sigma^*$, it holds that*
   (a) $\alpha \equiv \alpha$.
   (b) *If* $\alpha \equiv \beta$ *then* $\beta \equiv \alpha$.
   (c) *If* $\alpha \equiv \beta$ *and* $\beta \equiv \gamma$ *then* $\alpha \equiv \gamma$.

In subsection 2.1.5, it will become clear that it suffices to apply the cancellation rule in the forward direction (i.e., alternatively, that symmetry is not needed). Observe that $\alpha \equiv beta$ implies that for every $min\{0,1\}^*$, $\alpha(m) = \beta(m)$. The converse, to be referred to as the *freeness assumption*, states that the operator sequences satisfy only the identities implied by the cancellation rules. The freeness assumption plays a central role in the definition of insecurity (subsection 2.1.3).

2.1.2 What is a Ping-Pong Protocol

Intuitively, an instance of a ping-pong protocol is a sequence of operator words each applied in turn by a specified user. The "structure" of these sequences is predetermined by the protocol and the assignment of users to its "virtual" parties. We will first define this predetermined "structure" and only later consider its instances.

Definition 5 (protocols and protocol words): *A* var-operator f_x *consists of a form* $f \in F$ *and a variable* x *which assumes values in* I*. For* $p \geq 2$*, let* $\bar{x} = (x_1, x_2, ..., x_p)$ *be a sequence of* p *distinct variables assuming values in* I*, and* X *be the set of these variables.*
A p-party ping-pong protocol $P[\bar{x}]$ (over F) *is a sequence of pairs* $((y_1, \alpha_1[\bar{x}]), (y_2, \alpha_2[\bar{x}]), ..., (y_l, \alpha_l[\bar{x}]))$ *satisfying the following properties:*

1. *For every* $1 \leq j \leq l$*, it is the case that* $\alpha_j[\bar{x}]$ *is a sequence of var-operators indexed by variables in* X *(i.e.* $\alpha_j[\bar{x}] \in \{f_z : f \in F, z \in X\}^*$*).*
2. *For every* $1 \leq j \leq l$*, it is the case that* y_j *is one of the variables in* X *(i.e.* $y_j \in X$*).*
3. *For every* $1 \leq j \leq l$*, it is the case that* $\alpha_j[\bar{x}]$ *consists of operators in the vocabulary of* y_j*; that is,* $\alpha_j[\bar{x}] \in (\{f_z : f \in F \setminus F_D, z \in X\} \cup \{f_{y_j} : f \in F_D\})^*$*.*

$\alpha_j[\bar{x}]$ *is called the* j*-th* protocol word *of* $P[\bar{x}]$*.*

In the above definition, y_j $(= x_{i_j})$ indicates the party to be active in the j-th step and $\alpha_j[\bar{x}]$ determined (together with the assignment of users to $\bar{x}$) the operator word that this party will apply. Note that the protocol determines not only the forms of the operators in the α_j's, but also the relations between their indices.

Example 1: *Let* $F = \{E, D, a, d\}$ *and consider the two-party ping-pong protocol* $((x_1, E_{x_2}), (x_2, E_{x_1}D_{x_2}))$*. This protocol consists of two steps and has the following intuitive meaning. In the first step, the first party (the user assigned to play* x_1*) applies the encryption of the second party to the "initial message" and transmits the result. At the second step, the second party first applies its decryption and then applies the first party's encryption.*

The intuitive notion of an execution of a protocol can be given a precise formulation, by first defining an instance of a protocol.

Definition 6 (protocol instances): *Let* $\bar{u} = (u_1, u_2, ..., u_p)$ *be a sequence of* p *elements in* I*, and* $P[\bar{x}]$ *be a* p*-party ping-pong protocol as in Definition 5, and let* $i_1, i_2, ..., i_l$ *be the sequence of integers in* $\{1, 2, ..., p\}$ *satisfying* $y_j = x_{i_j}$*, for* $1 \leq j \leq l$*. That is,* $P[\bar{x}] = ((x_{i_1}, \alpha_1[\bar{x}]), ..., (y_{i_l}, \alpha_l[\bar{x}]))$*. An* $\bar{u}$-instance of the protocol $P[\bar{x}]$*, denoted* $P[\bar{u}]$*, is the result of substituting in* $P[\bar{x}]$ *the variable* x_j *by the user* u_j*, for every* $1 \leq j \leq p$*. The operator word* $\alpha_j[\bar{u}]$ *is called an* $\bar{u}$-instance of the protocol word $\alpha_j[\bar{x}]$*.*

Note that $P[\bar{u}]$ is a sequence of pairs $((u_{i_1}, \alpha_1[\bar{u}]), (u_{i_2}, \alpha_2[\bar{u}]), ..., (u_{i_l}, \alpha_l[\bar{u}]))$ such that $\alpha_j[\bar{u}]$ consists of operators in the vocabulary of u_{i_j}.

To specify an execution of a protocol, an initial message has also to be specified.

Definition 7 (protocol execution): *Let $\bar{u}$, P and $i_1, i_2, ..., i_l$ be as in Definition 6. Let $m_0 \in \{0,1\}^*$. An $\bar{u}$-*execution of $P[\bar{x}]$ on the message m_0 *is the following sequence of message transmissions:*

At the first step, u_{i_1} *transmits* $m_1 = \alpha_1[\bar{u}](m_0)$ *to* u_{i_2}.

For $j \in \{2, ..., l-1\}$, at the j-th step, u_{i_j} *transmits* $m_j = \alpha_j[\bar{u}](m_{j-1})$ *to* $u_{i_{j+1}}$.

At the last (i.e., l-th) step, u_{i_l} *transmits* $m_l = \alpha_l[\bar{u}](m_{l-1})$ *to all users in* $\bar{u}$.

m_0 *is called the* initial message *of this execution.*

It will be always assumed that m_0 is chosen by u_{i_1}. In the j-th step, u_{i_j} applies $\alpha_j[\bar{u}]$ to the message m_{j-1}. User u_{i_j} can do so, since $\alpha_j[\bar{u}]$ is over its vocabulary and it knows m_{j-1}. In case many instances of various protocols are played concurrently in the network, the users append to their transmissions a tag indicating the protocol they are playing ($P[\bar{x}]$), the instance ($\bar{u}$), and the step (j).

Given the definition of a p-party ping-pong protocol, *are the above definitions of its instances and executions acceptable*? We believe that the answer is **no** (or *not yet*). The above definitions include instances in which the same user plays the role of two different parties in the protocol (i.e. $\bar{u}$-instances in which the elements of $\bar{u}$ are not distinct). We feel that such executions are improper and should be excluded. If the protocol was designed for 3 parties it should be played by 3 distinct users and not by 2 users! We further assume that honest users will refuse to take part in improper executions of a protocol. The case in which this is not assumed is considered in subsection 2.4.2.

Definition 8 (proper execution): *Let $\bar{u} = (u_1, u_2, ..., u_p)$. An $\bar{u}$-instance ($\bar{u}$-execution) of a p-party ping pong protocol is called* proper *if $\bar{u}$ consists of p distinct elements in I.*

We end this subsection with an example of a proper execution.

Example 2: *Consider the two-party protocol of Example 1. Let $\bar{u} = (u_1, u_2)$, where $u_1 \neq u_2 \in I$. The following is a proper $\bar{u}$-execution of the protocol on the initial message m_0: user u_1 sends $m_1 = E_{u_2}(m_0)$ to u_2, and u_2 replies by sending $m_2 = E_{u_1}D_{u_2}(m_1) = E_{u_1}(m_0)$.*

2.1.3 The Definition of Insecurity

Referring to Example 2, *we ask whether a user $s \in I \setminus \{u_1, u_2\}$ can obtain* m_0? At first glance the answer seems negative, because only encrypted messages pass through the communication lines. But in second thought one may find a flaw which allows s to obtain the initial message m_0 as follows. User s intercepts the message m_1 and starts a (s, u_2)-instance of the protocol sending m_1 as the first transmission. Note that s does not know m_0 yet! But now, user u_2, following the (s, u_2)-instance sends $m' = E_s D_{u_2}(m_1)$ ($= E_s(m_0)$!) to s. Finally s applies D_s to m' and recovers m_0.

Before presenting the formal definition of insecurity, we motivate it by the following informal discussion of this notion. We will say that the protocol $P(\bar{x})$ is insecure if there exist an $\bar{u}$-execution of it such that an adversary s not in $\bar{u}$ can get the initial message through a fixed predetermined sequence of actions. The actions that the adversary s can take are of the following three types:

Type 1 – "passive" eavsedropping: Obtain (intercept) any message transmitted in the $\bar{u}$-execution.

Type 2 – local computations: Apply any operator in his vocabulary to any message. As a matter of fact user s can also ask other users (who collaborate with him) to apply operators in their vocabularies to any message. In the latter case, these users are knowingly collaborating with s and are certainly dishonest. We will assume that users in $\bar{u}$ will refuse to do so.

Type 3 – "active" attacks: Apply any **proper** $\bar{v}$-instance of any of the protocol's words to any message.

Let $\alpha_j[\bar{x}]$ be the j-th word of $P[\bar{x}]$. We first consider the case where $j > 1$. To obtain the affect of $\alpha_j[\bar{v}]$ on message m, user s waits for a $\bar{v}$-instance of P to occur (or convinces v_{i_1} to initiate such an instance on any message), replaces the $j-1$-st transmission by the desired message (m), and reads the j-th transmission. To obtain the affect of $\alpha_1[\bar{v}]$ on m, user s convinces user v_{i_1} to initiate a $\bar{v}$-execution on message m (and then s reads the first transmission).[3]

The users in $\bar{v}$ may unknowingly help user s, because they are playing according to the protocol and have no reason to suspect that their behavior helps someone to illegitimately obtain an initial message of another execution. In fact, we may assume that the user applying $\alpha_j[\bar{v}]$ (i.e., user v_{i_j}) is is unknowingly helping s (otherwise the affect of $\alpha_j[\bar{v}]$ could have been achieved by a sequence of Type 2 actions conducted by v_{i_j}). Since an honest user will refuse to take part in an improper instance, the $\bar{v}$-instance must be proper.

The above actions capture all that adversaries can do when knowing the protocol in use, but knowing nothing about the encryption functions used in implementing it. In other word, we may say that (as far as the notion of insecurity is concerned) it is assumed that the operators are "free" of any properties other than those algebraically implied by the cancellation rules (recall Definitions 3 and 4).

Definition 9 (insecurity): *Let $P[\bar{x}]$ be a p-party ping-pong protocol consisting of the words $\alpha_1[\bar{x}]$, $\alpha_2[\bar{x}]$,..., $\alpha_l[\bar{x}]$. Let $\bar{u} = (u_1, u_2, ..., u_p)$ be an arbitrary sequence of p distinct users and U be the set of these users.*

- *For every $J \subseteq I$, let Σ_J denote the union of the vocabularies of users in J; that is, $\Sigma_J = \cup_{j \in J} \Sigma_j$.*
- *For every $J \subseteq I$, let $\text{INST}(P, J)$ denote the set of all proper instances of protocol words of P in which the users are from J; that is,*

$$\text{INST}(P, J) = \{\alpha_j[\bar{v}] : 1 \le j \le l, \bar{v} = (v_1, v_2, ..., v_p) \text{ is proper and } v_1, v_2, ..., v_p \in J\}.$$

- *Protocol $P[\bar{x}]$ is* insecure *if there exist a set $S \subseteq I \setminus U$ and an operator string $\gamma \in (\Sigma_S \cup \text{INST}(P, S \cup U))^*$ such that $\gamma\alpha_1[\bar{u}] \equiv \lambda$.*

A few words of justification are in place. Firstly we note that, as shown in [DEK], the adversary $s \in S$ (trying to get the initial message m_0 in a $\bar{u}$-execution of P) may restrict its Type 1 actions to obtaining the first message transmitted in the $\bar{u}$-execution (i.e. $m_1 = \alpha_1[\bar{u}](m_0)$), because other Type 1 actions are covered by Type 3. Type 2 actions are fully captured by Σ_S, whereas Type 3

[3]The reader may wonder how is it possible to convince a user to initiate a v-execution on message m. This clearly requires a meta-protocol knowledge of the reasons and occasions in which users initiate executions. In any case, allowing s to commit such actions yields a stronger notion of security. Weaker notions can be obtained by restricting the actions of s; for example, allowing it to obtain the affect of $\alpha_j[\bar{v}]$ only for $j > 1$.

actions are captured by $\text{INST}(P, S \cup U)$. In case $\gamma\alpha_1[\bar{u}]$ is equivalent to the identity operator, user s can obtain the initial message m_0 by applying actions (corresponding to γ) to the message $\alpha_1[\bar{u}](m_0)$.

The users in S will be called *saboteurs.* The saboteurs are users not in $\bar{u}$ which help either knowingly or unknowingly in seizing the initial message.

Further comments: The choice of $\bar{u}$ in Definition 9 is immaterial: For every $\bar{u}, \bar{v}$, if a string γ exist (for demonstrating insecurity) with respect to the $\bar{u}$-instance then there exist a string δ (demonstrating insecurity) with respect to $\bar{v}$. Likewise, it is easy to verify that if the protocol's insecurity is demonstrated by a set of saboteurs S, then it can be demonstrated by any other set $S' \subseteq I \setminus U$ of the same cardinality (i.e. $|S| = |S'|$).

2.1.4 Extensions of the Insecurity Definition

The notion of insecurity can be extended for an "environment" (set) of protocols in the obvious manner. Instead of considering the instances of words of one protocol, we consider the instances of the words of all the protocols.

Definition $9E$ (insecurity of a set of protocols): *For $i = 1, .., q$, let p_i be an integer and P_i be a p_i-party ping-pong protocol with words $\alpha_{i,1}, \alpha_{i,2}, ..., \alpha_{i,l_i}$. The environment $\{P_i\}_{i=1}^{q}$ is* insecure *if there exist an $r \in \{1, 2, ..., q\}$, a p_r-long sequence $\bar{u}$, a set $S \subseteq I \setminus U$, and a string $\gamma \in (\Sigma_S \cup (\cup_{i=1}^{q} \text{INST}(P_i, S \cup U)))^*$ such that $\gamma\alpha_{r,1}[\bar{u}] \equiv \lambda$.*

All our (positive) results extend also to the security problem of environments.

Definition 9 (as well as Definition $9E$) can be rephrased as referring to the question *does there exist a string $\gamma \in (\Sigma_S \cup \text{INST}(P, S \cup U))^*$ that is equivalent to the left inverse of the first word in the protocol?* A natural generalization of the above question refers to the question *does there exist a string $\gamma \in (\Sigma_S \cup \text{INST}(P, S \cup U))^*$ that is equivalent to a specific operator word β?* In case the answer is positive, we will say that β is insecure in presence of the protocol P. Our results for the insecurity of p-party ping-pong protocols extend to the insecurity of operator words, indexed by at most p users, in presence of p-party ping-pong protocols. Further details are given in subsection 2.4.3.

2.1.5 Additional Terminology and Observations concerning Insecure Protocols

Recall that the insecurity of $P[\bar{x}]$ refers to the existemce of some sequence $\gamma \in (\Sigma_S \cup \text{INST}(P, S \cup U))^*$ having certain properties. The following definition refers to such a possible sequence.

Definition 10 ($\Delta(\cdot,\cdot,\cdot)$, insecurity strings, parsing and fillers): *Let $P[\bar{x}]$, $\bar{u}$, U, Σ_J and $\text{INST}(\cdot,\cdot)$ be as in Definition 9.*

1. *For $S \in I \setminus U$, we denote $\Sigma_S \cup \text{INST}(P, S \cup U)$ by $\Delta(P, U, S)$.*
2. *Let $\gamma \in \Delta(P, U, S)^*$ such that $\gamma\alpha_1[\bar{u}] \equiv \lambda$. Then $\gamma\alpha_1[\bar{u}]$ is called a S-*insecurity string *of $P[\bar{u}]$.*
3. *Let $\gamma = \gamma_n \cdots \gamma_2\gamma_1$ such that $\gamma_i \in \Delta(P, U, S)$, for $1 \le i \le n$. Then the sequence $(\gamma_1, \gamma_2, ..., \gamma_n)$ is called a* parsing[4] *of γ. An element γ_i* (of the parsing) *is called a* filler *if $\gamma_i \in \Sigma_S$.*

[4]Indeed, the parsing may not be unique.

Part 3 of Definition 10 describes the syntaxtic structure of insecurity strings as elements in the regular set $\Delta(P,U,S)^*\alpha_1[\bar{u}]$. We now describe insecurity strings as elements of the context-free grammer "generated" by the cancellation rules. To this end we first define a reduction process on operator words.

Definition 11 (reduction process): *Let $\delta_0 \in \Sigma^*$ be an operator word. A* reduction process on *δ_0 is a sequence of operator words $\delta_0, \delta_1, ..., \delta_t$ such that*

(1) *For every $i \in \{0,...,t-1\}$ there exists $\sigma, \tau \in \Sigma$ and $\alpha, \beta \in \Sigma^*$ such that (σ, τ) is a cancellation rule, $\delta_i = \alpha\sigma\tau\beta$, and $\delta_{i+1} = \alpha\beta$.*

In such a case, we say that σ and τ reduce each other in step i of the reduction process.

(2) *There exists no $\sigma, \tau \in \Sigma$ and $\alpha, \beta \in \Sigma^*$ such that (σ, τ) is a cancellation rule, $\delta_i = \alpha\sigma\tau\beta$, and $\delta_{i+1} = \alpha\beta$.*

The word δ_t is called the result of the reduction process.

Note that $\delta_0 \equiv \delta_1 \equiv \cdots \equiv \delta_t$.
Intuitively, a reduction process is a sequence of omissions of adjacent operators subject to a cancellation rule, resulting in a word that does not contain such adjacent operators. It is easy to see that the reduction system defined above is finite and has the Church Rosser property [CR,Ro]. By the Church Rosser Theorem, all reduction processes on δ yield the same result, called the reduced form of δ. Note that $\beta_1 \equiv \beta_2$ if and only if the reduced forms of β_1 and β_2 are identical.[5] In particular, the reduced form of an insecurity string is the empty string λ.

Definition 12 (cancellation pattern and mates): *Let $\delta_0 = \gamma\alpha_1[\bar{u}]$ be a S-insecurity string of $P[\bar{u}]$, and let $\delta_0, \delta_1, ..., \delta_t$ be a reduction process.*

- *We denote by $left(i)$ and $right(i)$ the locations in δ_0 of the operators that cancel each other in the i-th step of this reduction process; that is, $\delta_0 = \sigma_{2t}\cdots\sigma_{left(1)}\sigma_{right(1)}\cdots\sigma_1$, $\delta_i = \gamma_{i,2}\sigma_{left(i)}\sigma_{right(i)}\gamma_{i,1}$ and $\delta_{i+1} = \gamma_{i,2}\gamma_{i,1}$.*
- *For every $1 \le i \le t$, the locations $left(i)$ and $right(i)$* (in δ_0) *are called* mates. *The set of mates* (i.e. $\{(left(i), right(i))\}_{i=1}^t$) *is called the* cancellation pattern *of δ_0.*

Example 3: *Consider the following reduction sequence on $\delta_0 = \sigma_8\sigma_7\cdots\sigma_2\sigma_1$:*

$$\sigma_8\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1,\ \sigma_8\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1,\ \sigma_8\sigma_5\sigma_4\sigma_3,\ \sigma_4\sigma_3,\ \lambda.$$

The corresponding cancellation pattern is $\{(7,6),(2,1),(8,5),(4,3)\}$.

Conventions: We recall some of the conventions introduced so far.

(1) Whenever referring to a set of forms F, we implicitly consider its partition into the categories F_E, F_D, F_{app}, F_{delt}, F_{fun} and F_{inv}. Whenever we refer to a protocol we assume that the set of forms is fixed and understood from the context.

[5]Thus, the symmetry of $\equiv$ asserted in Definition 4 is note used.

(2) The set of users I is isomorphic to the sent of natural numbers, denoted N.

(3) Throughout the paper $U \subseteq I$ always denotes the set of users in $\bar{u}$.

(4) Whenever considering a S-insecurity string (as in Definition 10), we call the users in S saboteurs.

(5) Whenever considering an insecurity string we will refer to some fixed parsing and some fixed reduction sequence of it (as in Definition 11). Mates and the cancellation pattern (as in Definition 12) will be considered with respect to this fixed reduction sequence. Often we will present an insecurity string by a parsing of it.

(6) Sometimes, when presenting a protocol, we only present its protocol words. This is done when the protocol is only used to demonstrate some lower bound. In fact these words may belong to several protocols.

Two Simple Observations. The first observation is that it suffices to consider S-insecurity strings containing operators indexed by users in $U \cup S$. (Recall that Definition 10 talks of insecurity strings as sequences over $\text{INST}(P, U \cup S) \cup \Sigma_S$, whereas Σ_S contains non-decryption operators indexed by all users in I.)

Proposition 1: *If $P[\bar{u}]$ has a S-insecurity string then it also has a S-insecurity string consisting of operators in $\Sigma(F, U \cup S)$.*

Proof: Consider a S-insecurity string $\delta = \gamma\alpha_1[\bar{u}]$. If δ contains an operator indexed by a user **not in** $U \cup S$ then this operator is a filler (because it cannot be contained in a word in $\text{INST}(P, U \cup S)$ or in $\alpha_1[\bar{u}]$) and (for the same reason) its mate is also a filler. Let s be an arbitrary user in S. Replace in δ each filler $f_i \notin \Sigma(F, U \cup S)$ (i.e. $f \in F$ and $i \in I \setminus (U \cup S)$) by the filler f_s. The resulting string, denoted δ', has the same cancellation pattern as δ and the same non-fillers. Thus δ' is a S-insecurity string of $P[\bar{u}]$. ∎

In light of the above, we redefine $\Delta(P, U, S)$ to contain only words in $\Sigma(F, U \cup S)$; namely:

Definition $10'$ **(revised definition of $\Delta(\cdot,\cdot,\cdot)$ and insecurity strings):** *Let $P[\bar{x}]$, $\bar{u}$, U, Σ_J and $\text{INST}(\cdot,\cdot)$ be as in Definition 9.*

1. *For $S \in I \setminus U$, redefine $\Delta(P, U, S)$ to equal $(\Sigma_S \cap \Sigma(F, S \cup U)) \cup \text{INST}(P, S \cup U)$.*

   Note that $(\Sigma_S \cap \Sigma(F, S \cup U))$ contain only the operators that are indexed by users in $S \cup U$ and are in the vocabulary of users in S. That is,

$$(\Sigma_S \cap \Sigma(F, S \cup U)) = \{f_i : f \in F \setminus F_D\,,\, i \in S \cup U\} \cup \{f_s : f \in F_D\,,\, s \in S\}$$

2. *The string $\delta = \gamma\alpha_1[\bar{u}]$ is redefined to be a S*-insecurity string of $P[\bar{u}]$ *if $\gamma \in \Delta(P, U, S)^*$ and $\delta \equiv \lambda$. The point is that $\gamma \in (\Sigma_S \cap \Sigma(F, S \cup U)) \cup \text{INST}(P, S \cup U)^*$.*

   (*The parsing of δ is defined as in Definition 10.*)

The second observation is that with no loss of generality, we may assume that all non-fillers in an insecurity string are applied by non-saboteurs. Furthermore, these non-fillers must contain a decription operator indexed by a user in U.

Proposition 2: *Let $\delta = \gamma\alpha_1[\bar{u}]$ be a S-insecurity string of $P[\bar{u}]$. Consider a parsing of δ with minimum number of non-fillers. Then every non-filler contains an operator f_v such that $f \in F_D$ and $v \in U$.*

Proof: Let $\gamma\alpha_1[\bar{u}]$ be a S-insecurity string and $(\gamma_n, ..., \gamma_i, ..., \gamma_1)$ be a parsing of γ, with the minimum number of non-fillers. If γ_i is a non-filler containing no decryption operator of a user in U then $\gamma_i \in \Sigma_S^+$. It follows that $\gamma_i = \sigma_m \cdots \sigma_1$, where $\sigma_j in \Sigma_S$ for every $j \in \{1, ..., m\}$. Contradiction is reached by considering the sequence $(\gamma_n, ..., \gamma_{i+1}, \sigma_m, ..., \sigma_1, \gamma_{i-1}, ..., \gamma_1)$ which is an alternative parsing of γ with less non-fillers. ■

2.2 On the Number of Saboteurs

Definition 9 reduces the problem of testing the security of the multi-party ping-pong protocol $P[\bar{x}]$ to the following word problem: *Does there exist a set $S \subseteq I \setminus U$ and a string $\delta = \gamma\alpha_1[\bar{u}]$ such that $\gamma \in \Delta(P, U, S)^*$ and $\delta \equiv \lambda$?*

A natural question arises: *How large should the set S be so that for every insecure p-party ping-pong protocol $P[\bar{x}]$, there exist a S-insecurity string of $P[\bar{u}]$?* (Recall that the identity of the users is S is immaterial.) This question is of computational importance, because the cardinality of S determines the cardinality of $\Delta(P, U, S)$, which in turn effects the running time of the best algorithms known for solving the above word problem.[6]

Definition 13 ($\phi(\cdot)$ and $\Phi_F(\cdot)$):

- *For an* insecure *multi-party ping-pong protocol P, We denote by $\phi(P) = \phi(P[\bar{u}])$ the cardinality of the smallest set S such that there exists a S-insecurity string for $P[\bar{u}]$. In case the protocol $P[\bar{x}]$ is secure, $\phi(P)$ is undefined.*
- *Let F be a set of forms and p be an integer. We denote by $\Phi_F(p)$ the supremum of $\phi(\cdot)$, when taken over all* insecure *p-party ping-pong protocols that have forms in F.*

It is not assumed that $\Phi_F(p)$ is an integer. A-priori $\Phi_F : N \to N \cup \{\infty\}$. (It will be shown later that $\Phi_F(p)$ is an integer and so in fact $\Phi_F : N \to N$.) Let us begin our study of $\Phi_F(p)$ by recalling a result concerning the simple case of $p = 2$, due to Dolev, Even and Karp [DEK].

Proposition 3 [DEK]: *Let F be an arbitrary set of forms such that $F_E \neq \emptyset$. Then $\Phi_F(2) = 1$.*

Clearly, the case $F_E = \emptyset$ is of no interest.

[6]Specifically, let $P[\bar{x}]$ be a p-party ping-pong protocol, l denote the number of words in $P[\bar{x}]$ and n denote the sum of their corresponding length. Let $V_j^i = \frac{i!}{(i-j)!}$ denote the number of variations of j elements out of i elements. Then $|\Delta(P, U, S)| = |S| \cdot |F_D| + |U \cup S| \cdot |F \setminus F_D| + l \cdot V_p^{p+|S|} < O(1) + 2^{p+|S|} \cdot l$ and the total length of the strings in $\Delta(P, U, S)$ is $|S| \cdot |F_D| + |U \cup S| \cdot |F \setminus F_D| + n \cdot V_p^{p+|S|} < O(1) + 2^{p+|S|} \cdot n$. Recall that the running time of the Dolev-Even-Karp algorithm [DEK], which can be applied here too, is cubic in the total length of the strings in $\Delta(P, U, S)$. Thus, if $|S|$ can be bounded as a function of p (and $|F|$) then for constant p the latter expression is linear in n.

Proof: By Example 2 (subsection 2.1.2), $\Phi_F(2) \geq 1$. Consider an arbitrary insecure two-party ping-pong protocol $P[\bar{x}]$. Let $\delta = \gamma\alpha_1[\bar{u}]$ be a S-insecurity string of $P[\bar{u}]$, where $S \subseteq I \setminus \{u_1, u_2\}$ is an arbitrary set. By Proposition 2, *each non-filler $\gamma_i = \alpha_j[\bar{v}]$ in γ contains at least one operator indexed by a non-saboteur.* Therefore, out of the two elements of $\bar{v}$ at most one is in S. Let s be an arbitrary element in S. Replace in δ, each operator f_i with $i \in S$, by the operator f_s, resulting in a string δ'. The cancellation pattern of δ is maintained in δ', and all non-fillers in δ' are still proper instances of protocol words. Thus, δ' is a $\{s\}$-insecurity string of $P[\bar{u}]$. ■

We stress that the fact that the replacement preserves *properness* (as well as cancellation pattern), is due to the fact that the original insecurity string contains instances of protocol words with at most one saboteur. The later fact is implied by (Proposition 2 and) the fact that we considered only **two**-party protocols. For general p, we can only guarantee that each instance contains at most $p-1$ saboteurs. In the general case, "properness" will not be preserved by obvious replacements. This will be clarified in the coming subsections.

2.2.1 A Simple Case: $F = F_E \cup F_D$

In this subsection, we consider Φ_F in the special case where F contains only public-key cryptosystem forms (i.e., $F = F_E \cup F_D$). This special case is much simpler than the general case to be considered in the following subsections.

Theorem 1: *Let $F = F_E \cup F_D \neq \emptyset$. Then $\Phi_F(p) = p - 1$.*

Proof: We first demonstrate that $\Phi_{\{E,D\}}(p) \geq p-1$, by considering the following p-party ping-pong protocol, denoted $P[\bar{x}]$, which generalizes Example 1: $\alpha_1[\bar{x}] = E_{x_p}$ and $\alpha_2[\bar{x}] = E_{x_1}E_{x_2}\cdots E_{x_{p-1}}D_{x_p}$. The protocol $P[\bar{x}]$ is insecure, as can be demonstrated by the following $\{s_i\}_{i=1}^{p-1}$-insecurity string of $P[\bar{u}]$: $(D_{s_{p-1}}, ..., D_{s_2}, D_{s_1}, E_{s_1}E_{s_2}\cdots E_{s_{p-1}}D_{u_p}, E_{u_p})$. This allows us to write $\Phi_{\{E,D\}}(p) \geq \phi(P[\bar{x}])$. Turning to the analysis of $\phi(P[\bar{x}])$, let S be an arbitrary set such that $\gamma\alpha_1[\bar{u}]$ is a S-insecurity string of $P[\bar{u}]$. Evidently, $\gamma\alpha_1[\bar{u}]$ must contain as many occurrences of operators in $D_U = \{D_u : u \in U\}$ as occurrences of operators in $E_U = \{E_u : u \in U\}$. This implies that γ contains more occurrences of D_U-operators than of E_U-operators (because $\alpha_1[\bar{u}] = E_{u_p} \in E_U^+$). Since D_U-operators must appear in non-fillers, this may happen only if γ contains a $\bar{v}$-instance of $\alpha_2[\bar{x}]$, where $v_1, v_2, ..., v_{p-1}$ are elements in S and $v_p \in U$. Since $\alpha_2[\bar{v}]$ must be proper, the v_k's are distinct users. It follows that $\phi(P[\bar{x}]) \geq p-1$, and so $\Phi_{\{E,D\}}(p) \geq p-1$.

We now demonstrate that $\Phi_{F_E \cup F_D}(p) \leq p-1$, by considering an arbitrary *insecure* p-party ping-pong protocol ($P[\bar{x}]$) with forms in $F = F_E \cup F_D$. Recall $F_E = \{E^{(k)}\}_{k=1}^t$ and $F_D = \{D^{(k)}\}_{k=1}^t$. Let S be an arbitrary set and γ be a string such that $\delta = \gamma\alpha_1[\bar{u}]$ is a S-insecurity string of $P[\bar{u}]$. If $|S| \leq p-1$ then we are done. We thus consider the case in which $|S| > p-1$. By Proposition 2, all non-fillers in γ are activated by a user in U. Thus, only fillers may contain decryption operators indexed by a user in S (because $D_s^{(k)} \notin \Sigma_u$ when $s \in S$ and $u \in U$). By the above and since $F = F_E \cup F_D$, we have

> Fact: *if two operators indexed by a saboteur cancel each other then at most one of them* (i.e., the encryption) *occurs in a non-filler.*

Let $S' = \{s_i\}_{i=1}^{p-1}$ be an arbitrary set of $p-1$ users in S. We now replace in γ all occurrences of operators with index in S by operators (with the same form and) with index in S'. This replacement should be done with care so that "properness" of the word instances as well as the cancellation pattern are preserved. We iteratively consider all non-fillers in γ. With respect to each non-filler γ_i, we proceed in two steps, first replacing the non-filler itself and next replacing the mates of its saboteur-indexed operators. It is crucial that these mates are all fillers, and indeed this is guaranteed by the above Fact.

Step 1 – Replace a non-filler: Let $\gamma_i = \alpha_j[\bar{v}]$, and S_i be the set of users in $\bar{v}$ which are also in S. Recall that $|S_i| \leq p-1$ (because $\bar{v} = (v_1, v_2, ..., v_p)$ contains at least one user in U). Let $\theta_i : S_i \rightarrow S'$ be a one-to-one mapping, and $\bar{w} = (w_1, w_2, ..., w_p)$ be a sequence of p distinct users such that $w_k = v_k$ if $v_k \in U$ and $w_k = \theta_i(v_k)$ if $v_k \in S$. Such a mapping exist because $|S_i| \leq p-1$ and $|S'| = p-1$.

Action: replace in γ, the non-filler $\alpha_j[\bar{v}]$ by the non-filler $\alpha_j[\bar{w}]$.

Step 2 – Replace mate fillers: Let μ be a location in γ that changed its content from $E_v^{(k)}$ to $E_{\theta_i(v)}^{(k)}$ through the replacement of $\gamma_i = \alpha_j[\bar{v}]$ by $\alpha_j[\bar{w}]$ (note that the replacement affects only the index of operators in S and that their form, which is of encryption type, is preserved). Using the above notations, we note that μ's mate is a filler consisting of the operator $D_v^{(k)}$.

Action: change in γ the content of μ's mate from $D_v^{(k)}$ to $D_w^{(k)}$.

When the two steps concerning γ_i are completed, the resulting string is again an insecurity string (i.e., the properness and the cancellation pattern are preserved). When we are done with all non-fillers, the resulting string is a S'-insecurity string of $P[\bar{u}]$, demonstrating that $\phi(P[\bar{u}]) \leq p-1$. ■

Discussion: We stress that the fact that the presented replacement preserves the properness (as well as the cancellation pattern), is due to the fact that the original insecurity string does not contain operators indexed by saboteurs that cancel each other and are both occurring in non-fillers. This holds in case $F = F_E \cup F_D$, but may not hold in general! In general, the properness preservation requires that, in every non-filler, distinct saboteurs are replaced by distinct saboteurs. This induces inequality constraints on the replacement. The cancellation pattern preservation requires that mates' indices are replaced by the same saboteur. Also, the replacement *in each non-filler* must be consistent (i.e., two occurrences of the same saboteur in the same non-filler are replaced by the same saboteur). These induce equality constraints on the replacement. In both proofs of Proposition 3 and Theorem 1, it is easy to avoid conflicts between the equality and inequality constraints. In general, as we shortly illustrate, avoiding conflicts may be more difficult.

Example 4: *Consider the following $\{s_1, s_2, s_3\}$-insecurity string $(d_{s_2}d_{s_3}D_u, E_u a_{s_3} d_{s_1} D_u, E_u a_{s_1} a_{s_2})$. The cancellation pattern is necessarily $\{(3,4),(7,8),(2,5),(6,9),(1,10)\}$.*[7] *Each non-filler contains only 2 saboteurs. Nevertheless, it is impossible to replace (in the above insecurity string) the saboteurs (i.e., s_1, s_2 and s_3) by fewer than three saboteurs in a manner that will preserve both the properness and the cancellation pattern.*

[7]Recall that the operators in the insecurity string are indexed from right to left. The corresponding cancellation process is $d_{s_2}d_{s_3}D_uE_ua_{s_3}d_{s_1}D_uE_ua_{s_1}a_{s_2}$, $d_{s_2}d_{s_3}D_uE_ua_{s_3}d_{s_1}a_{s_1}a_{s_2} = d_{s_2}d_{s_3}D_uE_ua_{s_3}d_{s_1}\lambda\lambda a_{s_1}a_{s_2}$, $d_{s_2}d_{s_3}a_{s_3}d_{s_1}a_{s_1}a_{s_2} = d_{s_2}d_{s_3}\lambda\lambda a_{s_3}d_{s_1}\lambda\lambda a_{s_1}a_{s_2}$, $d_{s_2}d_{s_3}a_{s_3}a_{s_2} = d_{s_2}d_{s_3}\lambda\lambda a_{s_3}\lambda\lambda\lambda\lambda a_{s_2}$, $d_{s_2}a_{s_2} = d_{s_2}\lambda\lambda\lambda\lambda\lambda\lambda\lambda\lambda a_{s_2}$, λ.

2.2.2 The General Case: a Reduction to a Combinatorial Problem

As suggested by the above discussion, replacing saboteurs in insecurity strings is governed by specific inequality and equality constraints determined by the parsing and cancellation pattern of the string. However, the parsing and cancellation pattern of a string may not be unique, and furthermore every insecure protocol has infinitely many distinct insecurity strings. This makes the analysis of the number of saboteurs needed to demonstrate insecurity of p-party ping-pong protocols (i.e., $\Phi_F(p)$) very complicated. We chose to reduce the analysis of $\Phi_F(\cdot)$ to a much cleaner combinatorial problem that captures only the inequality and equality constraints (which may result from such insecurity strings). This subsection consists of the definition of the combinatorial problem and a reduction of $\Phi_F(p)$ to it.

Definition 14 (Well-Formed Parentheses Expression): *Let $\{[_i,]_i : i \in I\}$ be a set of parentheses symbols. The symbol $[_i$ is called a* left parenthesis (left-par)*, the symbol $]_i$ is called a* right parenthesis (right-par) *and both symbols are of the i-*th type*. Let $\gamma = \sigma_n \cdots \sigma_2 \sigma_1 \in \{[_i,]_i : i \in I\}^n$, for some natural number n. Then the μ-*th location in *γ contains σ_μ.*

- *The locations μ and ν, $\mu > \nu$, in γ are said to* match *if the σ_μ is an left-par and $\mu - \nu$ is the smallest integer such that $\sigma_\mu \sigma_{\mu-1} \cdots \sigma_{\nu+1} \sigma_\nu$ has an equal number of left-par's and right-par's.*
- *The locations μ and ν, $\mu > \nu$, in γ are said to be* type-matched *if they match and contain symbols of the same type. It follows that $\sigma_\mu = [_i$ and $\sigma_\nu =]_i$, for some i.*
- *$\gamma = \sigma_n \cdots \sigma_2 \sigma_1$ is a* well-form parentheses expression (abbreviated wfe) *if the integers 1 through n can be partitioned to type-matched pairs.*

Given a *wfe*, the matching of its locations is uniquely defined. This fact is very useful, as it frees us from having to specify the matching when presenting a *wfe*.

Definition 15 (regional wfe and regionwise-isomorphic wfe's): *Let $\gamma \in \{[_i,]_i : i \in I\}^*$ be a* wfe.

- *Let $\gamma_1, ..., \gamma_n \in \{[_i,]_i : i \in I\}$ be such that $\gamma = \gamma_n \cdots \gamma_2 \gamma_1$. Then $\bar{\gamma} = (\gamma_n, ..., \gamma_2, \gamma_1)$ is called a* regional wfe*, and each of the γ_i's is called a* region *of $\bar{\gamma}$.*
- *Let $\bar{\gamma} = (\gamma_n, ..., \gamma_2, \gamma_1)$ and $\bar{\delta} = (\delta_n, ..., \delta_2, \delta_1)$ be two regional* wfe*s. We say that $\bar{\gamma}$ and $\bar{\delta}$ are* regionwise-isomorphic *if for every $j \in \{1, ..., n\}$ the j-th region of $\bar{\gamma}$ i.e., γ_j) is* "isomorphic" *to the j-th region of $\bar{\delta}$ i.e., δ_j). That is, for every j, the following hold:*

  (1) *The length of γ_j equals the length of δ_j, which in turn is denoted l_j.*

  (2) *For every $1 \leq \mu \leq l_j$, the μ-th location in γ_j contains a left-par if and only if the μ-th location in δ_j contains a left-par.*

  (3) *For every $1 \leq \mu < \nu \leq l_j$, the symbols in locations μ and ν in γ_j are of the same type if and only if locations μ and ν in δ_j are of the same type.*

  That is, for every j, there exists a one-to-one mapping $\theta : I \rightarrow I$ such that for every μ if the μ-th location in γ_j contains $[_i$ (resp., *$]_i$) then the μ-th location in γ_j contains $[_{\theta(i)}$* (resp., *$]_{\theta(i)}$)*

Let $\bar{\gamma}$ and $\bar{\delta}$ be two regionwise-isomorphic *wfe*s. By condition (2) above, $\bar{\gamma}$ and $\bar{\delta}$ have the same matching pattern (i.e. if locations μ and ν in $\bar{\gamma}$ are matched then locations μ and ν in $\bar{\delta}$ are matched). Since both $\bar{\gamma}$ and $\bar{\delta}$ are *wfe*s, it is the case that matched locations are type-matched. Note the correspondence to the equality constraints on the replacement of saboteurs that are mates in an insecurity string. Also note the correspondence between condition (3) above and the constraints on the replacement of saboteurs in a non-filler (i.e., the inequality constraints are respected by the replacement). Clearly, the number of types appearing in corresponding regions of $\bar{\gamma}$ and $\bar{\delta}$ is identical. However, the number of types which appear in $\bar{\gamma}$ and $\bar{\delta}$ (as a whole) is not necessarily identical. This is demonstrated by the following regionwise-isomorphic *wfe*s: $\bar{\gamma} = ([_1,]_1 [_2,]_2 [_3 [_2,]_2]_3)$ and $\bar{\delta} = ([_2,]_2 [_1,]_1 [_2 [_1,]_1]_2)$. The second region in each of the above regional *wfe*s consists of symbols from two types (2 and 3 in $\bar{\gamma}$; 1 and 2 $\bar{\delta}$). However, $\bar{\gamma}$ contains 3 types and $\bar{\delta}$ contains only 2.

Definition 16 ($\psi(\cdot)$, q-regional wfe and $\Psi(\cdot)$): *Let $\bar{\gamma} = (\gamma_n, ..., \gamma_2, \gamma_1)$ be a regional* wfe*.*

- *We denote by $\psi(\bar{\gamma})$ the smallest integer m, such that there exists a regional* wfe *that is regional-isomorphic to $\bar{\gamma}$ and contains symbols of exactly m distinct types.*
- *We say that $\bar{\gamma}$ is a q-*regional wfe *(q-*wfe*) if each region of $\bar{\gamma}$ contains symbols of at most q distinct types.*

We denote by $\Psi(q)$ the supremum of $\psi(\cdot)$, when taken over all q-regional wfe*s.*

The intuitive correspondence between $\Phi_F(\cdot)$ and $\Psi(\cdot)$ is formulated in the following two Lemmas:[8]

Lemma 1: *For every $p \geq 2$ and F, it holds that $\Psi(p-1) \geq \Phi_F(p)$.*

Lemma 2: *For every $q \geq 1$ and F such that $F_E \neq \emptyset$ and $F_{app} \neq \emptyset$, it holds that $\Phi_F(q+1) \geq \Psi(q)$.*

As an immediate corollary, we get

Theorem 2: *For every $p \geq 2$ and F such that $F_E \neq \emptyset$ and $F_{app} \neq \emptyset$, it holds that $\Phi_F(p) = \Psi(p-1)$.*

We now restate and prove the above Lemmas.

Lemma 1 (restated): *Let $P[\bar{x}]$ be an insecure p-party ping-pong protocol. Then there exist a $(p-1)$-regional* wfe *$\bar{\delta}$ such that $\psi(\bar{\delta}) = \phi(P[\bar{u}])$.*

Assuming that $\Phi_F(p)$ is finite, let P be a p-party protocol such that $\phi(P[\bar{u}]) = \Phi_F(p)$. It follows that $\Psi(p-1) \geq \psi(\bar{\delta}) = \phi(P[\bar{u}]) = \Phi_F(p)$. In case $\Phi_F(p)$ is infinite we consider, for every $n \in N$, a p-party protocol P such that $\phi(P[\bar{u}]) \geq n$ and obtain $\Psi(p-1) \geq \psi(\bar{\delta}) = \phi(P[\bar{u}]) \geq n$.

[8]In the formulation, we allow both $\Psi(q)$ and $\Phi_F(p)$ to be infinite, with $n < \infty \leq \infty = \infty$ for every natural number n.

Proof: The lemma follows by a natural transformation of insecurity strings into regional *wfe*s. One needs only stress that all non-fillers in the insecurity string contain at most $p-1$ saboteurs. Following are the details of the transformation.

Let $\gamma\alpha_1[\bar{u}]$ be a S-insecurity string for $P[\bar{u}]$ such that $|S| = \phi(P[\bar{u}])$, and let $\bar{\gamma} = (\gamma_n, ..., \gamma_2, \gamma_1)$ be a parsing of γ. By Proposition 2, each γ_i contains at most $p-1$ saboteurs. Using $\bar{\gamma}$, we construct a *wfe* $\bar{\delta} = (\delta_n, ..., \delta_2, \delta_1)$ as follows:

(1) *Omit* from $\bar{\gamma}$ all operators with index in U, resulting in a sequence $\bar{\beta}$.

Locations in $\bar{\beta}$ are *paired* by their correspondence to mates in the reduction process of $\gamma\alpha_1[\bar{u}]$.

(2) Let (μ, ν) be a pair of locations in $\bar{\beta}$ that correspond to mates in $\bar{\gamma}$. Then, there exist $s \in S$ and $f, g \in F$ such that location μ contains the operator f_s while location ν contains g_s. Suppose that $mu > nu$.

Replace in location μ the operator f_s by the symbol $[_s$, and in location ν the operator g_s by the symbol $]_s$.

Applying this to all location-pairs yields a regional *wfe*, denote $\bar{\delta}$.

Note that $\bar{\delta}$ is a $(p-1)$-regional *wfe*, which fully captures the parsing and the reduction process of the saboteur operators in the insecurity string $\gamma\alpha_1[\bar{u}]$. The inequality and equality constraints on the replacement of saboteurs in $\gamma\alpha_1[\bar{u}]$ are fully captured by corresponding constraints on *wfe*s that are regional-isomorphic to $\bar{\delta}$.

Claim: $\psi(\bar{\delta}) = \phi(P[\bar{u}])$.

Proof: By its construction, $\bar{\delta}$ contains $\phi(P[\bar{u}])$ distinct types, and so $\psi(\bar{\delta}) \leq \phi(P[\bar{u}])$. Assume that $\bar{\delta}$ has a regionwise-isomorphic *wfe* $\bar{\delta}'$ with types in S', where $|S'| < |S|$. The correspondence, between saboteur indexed operators in $\bar{\gamma}$ and symbols in $\bar{\delta}$, can be used to derive from $\bar{\delta}'$ a replacement of the saboteurs in $\bar{\gamma}$. Let $\bar{\gamma}'$ be the parsing resulting by this replacement. The reader can easily verify that $\gamma'\alpha_1[\bar{u}]$ is a S'-insecurity string of $P[\bar{u}]$ with the same parsing and cancellation pattern as $\gamma\alpha_1[\bar{u}]$. This yields $\phi(P[\bar{u}]) \leq |S'| < |S|$ in contradiction to the hypothesis $|S| = \phi(P[\bar{u}])$. □

The Lemma follows. ■

Lemma 2 (restated): *Let E, D, a and d be encryption, decryption, name-appending and name-deletion forms, respectively. Let $\bar{\delta}$ be a q-regional* wfe*. Then there exist an insecure $(q+1)$-party ping-pong protocol $P[\bar{x}]$ over the forms $\{E, D, a, d\}$ such that $\phi(P[\bar{u}]) = \psi(\bar{\delta})$.*

Assuming that $\Psi(q)$ is finite, let $\bar{\delta}$ be a q-regional *wfe* such that $\psi(\bar{\delta}) = \Psi(q)$. It follows that $\Phi_{\{E,D,a,d\}}(q+1) \geq \phi(P) = \psi(\bar{\delta}) = \Psi(q)$. The case in which $\Psi(q)$ is infinite is handled similarly.

Proof: The transformation of regional *wfe*s into insecurity strings is more difficult than the converse transformation presented in the proof of Lemma 1. One needs to construct an insecure protocol such that each of its insecurity strings can be associated to the given regional *wfe*.

We first use $\bar{\delta}$ to construct a $(q+1)$-party ping-pong protocol $P[\bar{x}]$ (over $\{E, D, a, d\}$), next show that $P[\bar{x}]$ is insecure (and $\phi(P[\bar{u}]) \leq \psi(\bar{\delta})$), and end by proving that $\phi(P[\bar{u}]) \geq \psi(\bar{\delta})$.

The Construction of the Protocol: Without loss of generality, $\bar{\delta} = (\delta_n, ..., \delta_2, \delta_1)$ contains symbols of the types $T = \{1, 2, ..., \psi(\bar{\delta})\}$. Let $T_i \subseteq T$ denote the set of types occurring in δ_i, and $\bar{x} = (x_1, x_2, ..., x_q, x_{q+1})$ be a sequence of $q + 1$ distinct variables. Let $\theta_i : T_i \to \{x_1, x_2, ..., x_q\}$ be a one-to-one mapping. Such mappings exist since $|T_i| \leq q$. We proceed as follows:

Construct protocol sub-words that correspond to $\bar{\delta}$: For every $i = 1, ..., n$ and $j \in T_i$, we *replace* in δ_i the symbol $[_j$ by the operator $d_{\theta_i(j)}$, and the symbol $]_j$ by the operator $a_{\theta_i(j)}$. The result is a var-operator word, denoted $\beta_i^{(W)}[\bar{x}]$, in $\{a_{x_i}, d_{x_i} : 1 \leq i \leq q\}^*$.

Define auxiliary sub-words: For every $i \in \{0, 1, ..., n+1\}$, define $\beta_i^{(D)}[\bar{x}] = d_{x_{q+1}} E_{x_{q+1}} (d_{x_{q+1}})^i D_{x_{q+1}}$, and $\beta_i^{(A)}[\bar{x}] = E_{x_{q+1}} (a_{x_{q+1}})^{i+1} D_{x_{q+1}} a_{x_{q+1}}$. Note that all operators in $\beta_i^{(D)}[\bar{x}]$ and $\beta_i^{(A)}[\bar{x}]$ are indexed by x_{q+1}.

Define $\beta^{(S)}[\bar{x}] = E_{x_1} E_{x_2} \cdots E_{x_q}$, and note that $\beta^{(S)}[\bar{x}]$ is in the vocabulary of each user.

The protocol itself: The $(q + 1)$-party ping-pong protocol $P[\bar{x}]$ consists of the following words:

- $\alpha_1[\bar{x}] = \beta_0^{(A)}[\bar{x}]$.
- For $i = 1, ..., n$, $\alpha_{i+1}[\bar{x}] = \beta^{(S)}[\bar{x}] \beta_i^{(A)}[\bar{x}] \beta_i^{(W)}[\bar{x}] \beta_i^{(D)}[\bar{x}]$.
- $\alpha_{n+2}[\bar{x}] = \beta_{n+1}^{(D)}[\bar{x}]$.

The protocol words are constructed so that they may appear in a minimal length insecurity string only in consequtive order. This will be argued later. Let us first show that $P[\bar{x}]$ is insecure.

The Insecurity of the Protocol: Let $S = \{1, 2, ..., \psi(\bar{\delta})\} = T$ and $\bar{u} = (\psi(\bar{\delta})+1, \psi(\bar{\delta})+2, ..., \psi(\bar{\delta})+q+1)$. Let R_i be an arbitrary subset of $S \setminus T_i$ having cardinality $q - |T_i|$. Let $\theta_i^{-1} : \{x_1, x_2, ..., x_q\} \to (T_i \cup R_i)$ be a one-to-one mapping such that $\theta_i^{-1}(\theta_i(j)) = j$ for every $j \in T_i$. Let $\theta_i^{-1}(\bar{x}) = (\theta_i^{-1}(x_1), \theta_i^{-1}(x_2), ..., \theta_i^{-1}(x_q), \psi(\bar{\delta}) + q + 1)$. The following facts can be easily verified.

Fact 1: For every $i \in \{0, 1, ..., n\}$ and every $\bar{v}$ and $\bar{w}$ such that $v_{q+1} = w_{q+1}$, it holds that $\beta_{i+1}^{(D)}[\bar{v}] \beta_i^{(A)}[\bar{w}] \equiv \lambda$.

Proof: By its definition $\beta_{i+1}^{(D)}[\bar{v}] \beta_i^{(A)}[\bar{w}] = d_{v_{q+1}} E_{v_{q+1}} (d_{v_{q+1}})^{i+1} D_{v_{q+1}} \cdot E_{w_{q+1}} (a_{w_{q+1}})^{i+1} D_{w_{q+1}} a_{w_{q+1}}$, which in turn is equivalent to $d_{v_{q+1}} E_{v_{q+1}} (d_{v_{q+1}})^{i+1} (a_{w_{q+1}})^{i+1} D_{w_{q+1}} a_{w_{q+1}} \equiv d_{v_{q+1}} E_{v_{q+1}} D_{w_{q+1}} a_{w_{q+1}} \equiv d_{v_{q+1}} a_{w_{q+1}} \equiv \lambda$. $\Box$

Fact 2: The operator string $\beta_n^{(W)}[\theta_n^{-1}(\bar{x})] \cdots \beta_2^{(W)}[\theta_2^{-1}(\bar{x})] \beta_1^{(W)}[\theta_1^{-1}(\bar{x})]$ is equivalent to the empty string λ.

Proof: The said operator string is isomorphic to $\bar{\delta}$, where the isomorphism maps append/delete operators that are indexed by elements in T to corresponding right/left parenthesis. The fact that $\bar{\delta}$ is a *wfe* implies the desired reduction process. $\Box$

We now extend the "main" protocol words by decryption operators that are indexed by the first q parties (to be played by saboteurs): For $i = 1, ..., n$, let $\alpha_{i+1}^{(ext)}[\bar{v}] = D_{v_q} \cdots D_{v_2} D_{v_1} \alpha_{i+1}[\bar{v}]$. Note that $\alpha_{i+1}^{(ext)}[\bar{v}] \equiv \beta_i^{(A)}[\bar{v}] \beta_i^{(W)}[\bar{v}] \beta_i^{(D)}[\bar{v}]$, and that $\alpha_{i+1}^{(} ext)[\theta_i^{-1}(\bar{x})] \in \Delta(P, U, S)^*$. Let $v = \theta_i^{-1}(\bar{x})$ and

$\gamma = \alpha_{n+2}[\bar{v}]\alpha_{n+1}^{(ext)}[\bar{v}] \cdots \alpha_2^{(ext)}[\bar{v}]$. Note that $v_{q+1} = \psi(\bar{\delta}) + q + 1 = u_{q+1}$. Using the above definitions and Fact 1, we have

$$\begin{aligned}
\gamma\alpha_1[\bar{u}] &\equiv \beta_{n+1}^{(D)}[\bar{v}] \cdot \beta_n^{(A)}[\bar{v}]\beta_n^{(W)}[\bar{v}]\beta_n^{(D)}[\bar{v}] \cdot \beta_{n-1}^{(A)}[\bar{v}]\beta_{n-1}^{(W)}[\bar{v}]\beta_{n-1}^{(D)}[\bar{v}] \cdots \beta_1^{(A)}[\bar{v}]\beta_1^{(W)}[\bar{v}]\beta_1^{(D)}[\bar{v}] \cdot \beta_0^{(A)}[\bar{u}] \\
&\equiv \beta_n^{(W)}[\bar{v}]\beta_{n-1}^{(W)}[\bar{v}] \cdots \beta_1^{(W)}[\bar{v}]
\end{aligned}$$

which is equivalent to λ (by Fact 2). It follows that $\gamma\alpha_1[\bar{u}]$ is an S-insecurity string of $P[\bar{u}]$ and $\phi(P[\bar{u}]) \leq \psi(\bar{\delta})$.

Lower Bound on Number of Saboteurs (in insecurity strings): We will now show that $\phi(P[\bar{u}]) \geq \psi(\bar{\delta})$. Let S' be an arbitrary set such that $\gamma = (\gamma_m, ..., \gamma_2, \gamma_1)$ is a S'-insecurity string of P; that is, $\gamma\alpha_1[\bar{u}] \equiv \lambda$. The following facts concerning γ are of interest.

New Note: *It is instructive to have a schematic picture of a generic $\alpha_{j+1}[\bar{v}]$, which (for $j \leq n$) equals $\beta^{(S)}[\bar{v}]\beta_j^{(A)}[\bar{v}]\beta_j^{(W)}[\bar{v}]\beta_j^{(D)}[\bar{v}]$ if $j \leq n$. Such picture has the form*

S-block	*A-block*	*W-block*	*D-block*

Fact 3 (resp., Fact 4) asserts that the mates of the operators in the D-block (resp., A-block) must lie to its right (resp., left).

Fact 3: For any $i \in \{1, ..., m\}$, let $\gamma_i = \alpha_{j+1}[\bar{v}]$ be a non-filler, for some $j \in \{1, ..., n+1\}$. Consider the $\beta_j^{(D)} = d_{v_{q+1}}E_{v_{q+1}}(d_{v_{q+1}})^j D_{v_{q+1}}$ part of $\alpha_{j+1}[\bar{v}]$ (which, in turn, equals $\beta^{(S)}[\bar{v}]\beta_j^{(A)}[\bar{v}]\beta_j^{(W)}[\bar{v}]\beta_j^{(D)}[\bar{v}]$ if $j \leq n$). Then the mates of operators in this part lie to its right in γ.

Proof: Observe that the leftmost operator in $\beta_j^{(D)}$ is a name-deletion operator, which must have a mate on its right. □

Fact 4: Let $\gamma_i = \alpha_{j+1}[\bar{v}]$ be a non-filler, for some $j \in \{0, ..., n\}$. Consider the $\beta_j^{(A)} = E_{v_{q+1}}(a_{v_{q+1}})^{j+1}D_{v_{q+1}}a_{v_{q+1}}$ part of $\alpha_{j+1}[\bar{v}]$. Then the mates of operators in this part lie to its left in γ.

Proof: Observe that the rightmost operator in $\beta_j^{(A)}$ is a name-appending operator, which must have a mate on its left. □

New Note: *It seems that the order of Facts 5 and 6 can be switched, resulting in a more natural order.*

Fact 5: Let $\beta_1, \beta_2, \beta_3$ be arbitrary operator strings that do not contain any name-appending/deletion operators. Let v and w be arbitrary users. Suppose that $\beta_1 d_v \beta_2 a_w \beta_3 \equiv \lambda$. Then $\beta_2 \equiv \lambda \equiv \beta_1\beta_3$ and $v = w$.

Proof: Observe that a_w has a mate to its left, d_v has a mate on its right, and the "mating relation" constitutes a "well formed parentheses expression". Thus, $v = w$, $\beta_2 \equiv \lambda$ and $\beta_1\beta_3 \equiv \lambda$. □

Fact 6: For any $i \in \{1, ..., m\}$, let $\gamma_i = \alpha_{j+1}[\bar{v}]$ be a non-filler, for some $j \in \{1, ..., n\}$. Then, for every $k \in \{1, ..., q\}$, it holds that $v_k \in S'$.

Proof: Assume, to the contrary that $v_k \in U$. Then one of the operators in the corresponding $\beta^{(S)}[\bar{v}]$ is an encryption by a user in U and must have a mate in a non-filler (because the mate is a decryption). The structure of α_{j+1} (and Fact 4) forces this encryption to have a mate on its left. By Fact 3, this mate is in the $\beta_l^{(D)}$ part of some $\gamma_{i'} = \alpha_{l+1}[\bar{w}]$. It follows that $w_{q+1} = v_k$ and $D_{w_{q+1}}\gamma_{i'-1}\cdots\gamma_{i+1}E_{v_1}\cdots E_{v_k} \equiv \lambda$. But it also follows that $d_{w_{q+1}}D_{w_{q+1}}\gamma_{i'-1}\cdots\gamma_{i+1}E_{v_1}\cdots E_{v_k}E_{v_{k+1}}\cdots E_{v_q}E_{v_{q+1}}a_{v_{q+1}} \equiv \lambda$. But this implies $w_{q+1} = v_{q+1}$, which violates the properness of $\gamma_i = \alpha_{j+1}[\bar{v}]$. $\Box$

Fact 7: Without loss of generality, if $\gamma_i = \alpha_{j+1}[\bar{v}]$ then $\gamma_{i+k} = D_{v_k}$ for every $1 \leq k \leq q$. Here, "without loss of generality" means that there exist a S'-insecurity string γ for which the claim holds.

Proof: Using Fact 6, one can always substitute in γ the non-filler $\alpha_{j+1}[\bar{v}]$ (which is in $\Delta(P,U,S')$) by the word-sequence $E_{v_1}E_{v_2}\cdots E_{v_q}D_{v_q}\cdots D_{v_2}D_{v_1}\alpha_{j+1}[\bar{v}] \in \Delta(P,U,S')^*$. (The key observation is that by Fact 6, $v_1,...,v_q \in S'$, and thus $D_{v_1},...,D_{v_q} \in \Delta(P,U,S')$.) $\Box$

Thus, we may consider γ as consisting of fillers and words in $\{\alpha_{j+1}^{(ext)}[\bar{v}] : 1 \leq j \leq n\} \cup \{\alpha_1[\bar{v}], \alpha_{n+2}[\bar{v}]\}$. Note that such a partition does not correspond to the formal definition of a parsing, but we will use it nevertheless.

Fact 8: Without loss of generality, for $j = 1,...,n$, it holds that $\gamma_j = \alpha_{j+1}^{(ext)}[\cdot]$. Also, $\gamma_{n+1} = \alpha_{n+2}[\cdot]$ and $\gamma_{n+1}\cdots\gamma_1\alpha_1[\bar{u}] \equiv \lambda$.

Proof: We prove the claim by induction on j. For the basis case ($j = 1$), note that $\alpha_1[\bar{u}] = \beta_0^{(A)}[\bar{u}] = E_{u_{q+1}}a_{u_{q+1}}D_{u_{q+1}}a_{u_{q+1}}$ must be cancelled by some $\beta_{k+1}^{(D)}[\bar{w}]$ (use Facts 3 and 4), and $w_{q+1} = u_{q+1}$ holds. Considering the possible cancellation pattern in $\beta_{k+1}^{(D)}[\bar{w}]\cdots\alpha_1[\bar{u}] = d_{u_{q+1}}E_{u_{q+1}}(d_{u_{q+1}})^{k+1}D_{u_{q+1}}\cdots E_{u_{q+1}}a_{u_{q+1}}D_{u_{q+1}}a_{u_{q+1}}$, one infers that $k = 0$. Furthermore, we may replace the word between $\beta_{k+1}^{(D)}[\bar{w}]$ and $\alpha_1[\bar{u}]$ by λ. Similarly (in the induction step), $\beta_j^{(A)}$, which appears in $\gamma_j = \alpha_{j+1}^{(ext)}[\cdot]$ must be cancelled by some $\beta_{i+1}^{(D)}[\cdot]$. The same reasoning also establishes $\gamma_{n+1} = \alpha_{n+2}[\cdot]$ and $\gamma_{n+1}\cdots\gamma_1\alpha_1[\bar{u}] \equiv \lambda$. $\Box$

Fact 9: Let $\bar{v}^{(i)}$ be the users in the i-th non-filler. Then, $\beta_n^{(W)}[\bar{v}^{(n)}]\cdots\beta_2^{(W)}[\bar{v}^{(2)}]\beta_1^{(W)}[\bar{v}^{(1)}] \equiv \lambda$.

Proof: By Fact 8, it holds that $\alpha_{n+2}^{(ext)}[v^{(n+1)}]\cdots\alpha_2^{(ext)}[v^{(1)}]\alpha_1[\bar{u}] \equiv \lambda$. By the structure of the α_j's, it follows that $\beta_n^{(W)}[\bar{v}^{(n)}]\cdots\beta_2^{(W)}[\bar{v}^{(2)}]\beta_1^{(W)}[\bar{v}^{(1)}] \equiv \lambda$. $\Box$

By Facts 6 and 9, there exist $\bar{v}^{(i)} = (v_1^{(i)}, v_2^{(i)}, ..., v_{q+1}^{(i)})$'s such that $\beta_n^{(W)}[\bar{v}^{(n)}]\cdots\beta_2^{(W)}[\bar{v}^{(2)}]\beta_1^{(W)}[\bar{v}^{(1)}] \equiv \lambda$ and $v_j^{(i)} \in S'$, for every $i \in \{1,...,n\}$ and $j \in \{1,...,q\}$. Using the correspondance between $\bar{\delta}$ and $(\beta_n^{(W)},...,\beta_2^{(W)},\beta_1^{(W)})$, we can obtain a regionwise-isomorhich $\bar{\delta}'$ that contains $|S'|$ symbols. By taking S' such that $\phi(P) = |S'|$ and considering a S'-insecurity string of P, we conclude that $\psi(\bar{\delta}) \leq |S'| = \phi(P)$. The lemma follows. ■

Discussion: The reduction of $\Phi_F(p)$ to $\Psi(p-1)$ may not seem a dramatic simplification, but consider the analysis of $\phi(P)$ for the worst p-party protocol P versus the analysis of $\psi(\bar{\delta})$ for the worst $\bar{\delta}$. In the former case we need to consider all insecurity strings of P (as well as all their parsings and cancellation pattern), whereas in the latter case we need only consider the "type-colorings" of one fixed string.

Example 5: *Consider first the problem of determining $\Phi_F(2)$ versus the problem of determining $\Psi(1)$. In the latter case, we refer to all possible 1-regional* wfe*s, and it is clear that each such* wfe *can be "type-colored" by one set of parenthesis. Thus,* $\Psi(1) = 1$ (and $\Phi_F(2) = 1$ follows). *Furthermore, to show that $\Phi_F(p) > p-1$ for some p, it suffices to show that $\Psi(q) > q$ for some $q = p-1$. This can be shown, for $q = 2$, by considering the 2-regional* wfe $\bar{\delta} = ([_1[_2,]_2[_3,]_3[_1)$. *Clearly, $\psi(\bar{\delta}) = 3$, and $\Psi(2) \geq 3$ follows.*

2.2.3 Lower Bounds on the Combinatorial Problem

We have seen in Example 5 that $\Psi(2) \geq 3$, which improves over the trivial lower bound of $\Psi(2) \geq 2$. The argument can be easily extended to yield $\Psi(2q') \geq 3q'$ (e.g., by replacing each par-symbol by q' different symbols). In this section we further investigate the non-triviality of of Ψ establishing $\Psi(q) \geq 3q-2$ for every $q \geq 1$ (e.g., $\Psi(1) \geq 1$, $\Psi(2) \geq 4$ and $\Psi(3) \geq 7$). It is even not a priori clear whether $\Psi(q)$ is at all finite, for $q \geq 2$. The task of providing an upper bound on $\Psi(q)$ is undertaken in section 2.2.4.

New Note: *Translating the results in [EG] to the current terminology we have:*

Lemma 3 in [EG] (p. 16)**:** $\Psi(2) \geq 5$.

Lemma 4 in [EG] (p. 16)**:** $\Psi(q) \geq 3q-2$, *for every* $q \geq 1$.

Both claims are proven by presenting relatively simple regional wfe's.

We comment that the presentation in [EG] takes another step of abstraction: When considering a q-regional wfe, each occurrence of a symbol of type i (regardless of whether it is a left-par or a right-par) is substituted by a variable x_i, and we consider assignments to the variables such that different variables that appear in the same region are not assigned the same value. The question is how many values must be used in such an assignment.

2.2.4 Upper Bounds on the Combinatorial Problem

New Note: *Translating the results in [EG] to the current terminology we have:*

Lemma 6 in [EG] (p. 18)**:** $\Psi(q) \leq 3q-1$, *for every* $q \geq 1$.

This lemma is proven in Appendix B of [EG] (pp. 33–45). Note that it follows that $\Psi(2) = 5$ and $\Psi(q) \in \{3q-2, 3q-1\}$, for every $q \geq 3$. Recall that $\Psi(1) = 1$.

In continuation to the note in Section 2.2.3, we comment that the assignment problem can be cast as a coloring problem of a corresponding graph in which variables are represented by vertices and edges represent pairs of variables that appear in the same region. The presentation in [EG] is in therse terms.

2.3 NP–Hardness for the Case of Varying Number of Parties

New Note: *This is proven in [EG] by reduction from a restricted form of 3XC, which is shown to be NP-complete (by reduction from 3XC), to the problem of determining the security of multi-party ping-pong protocols over the set of forms $F = \{E, D, a, d\}$. See pages 18–22 of [EG].*

2.4 Further Discussions Concerning the Insecurity Definition

In this section we discuss four side issues concerning the insecurity definition of section 2.1.

2.4.1 Other Categories of Operator-Forms

The operator-forms considered in subsection 2.1.1, were placed in (three pairs of) categories according to their role in the users vocabularies and in the cancellation rules. Let (f, g) be a pair of forms. We say that f and g cancel symmetrically if, for every $i \in I$, both (f_i, g_i) and (g_i, f_i) are cancellation rules. The forms f and g cancel asymmetrically if, for every $i \in I$, only (f_i, g_i) is a cancellation rule. We say that f is public if for every $i, j \in I$, the operator f_i is in j's vocabulary. The form f is private if $f_i \in \Sigma_j$ implies $i = j$. A systematically enumeration of all "reasonable" pairs of categories can be found in the following table.

Cancellation pattern / Form's scope	symmetrically	asymmetrically
both public	F_{fun} and F_{inv}	F_{delt} and F_{app}
one private	F_E and F_D	Cases 1 and 2 (below)
both private	Case 4 (below)	Case 3 (below)

We first consider the first three new cases (in the table above), where the forms cancel asymmetrically. Let (L, R) denote such a pair of forms, where (L_i, R_i) is a cancellation rule, for every $i \in I$. The three cases we consider are:

Case 1: L is private and R is public.

This case corresponds to a public-key encryption system that (unlike the basic Diffie and Hellman model [DH]) cannot be directly used for digital signatures (e.g., see [GM]). We can model these forms by the categories of subsection 2.1.1, by letting $L_u = d_u D_u$ and $R_u = E_u a_u$, where $E \in F_E$, $D \in F_D$, $a \in F_{app}$ and $d \in F_{delt}$ are special forms not used for other purposes. Testing security of protocols over $\Sigma(F' \cup \{L, R\}, I)$ can be reduced to testing security of protocols over $\Sigma(F' \cup \{E, D, a, d\}, I)$.

Case 2: L is public and R is private.

This case corresponds to a public-key signature scheme that cannot be directly used for encryption. Again, we can model these forms by the categories of subsection 2.1.1, by letting $L_u = d_u E_u$ and $R_u = D_u a_u$, where $E \in F_E$, $D \in F_D$, $a \in F_{app}$ and $d \in F_{delt}$ are special forms not used for other purposes. A similar reduction of the security problem holds.

Case 3: Both L and R are private.

This case corresponds to a private message authentication scheme. Modeling is done by letting $L_u = L_u^{(1)} L_u^{(2)}$ and $R_u = R_u^{(2)} R_u^{(1)}$, where $(L^{(1)}, R^{(1)})$ is a forms-pair of the Case 1 (above) and $(L^{(2)}, R^{(2)})$ is a pair of the Case 2.

Finally, we get to Case 4: We consider a pair of forms (f, f^{-1}) where f and f^{-1} are both private and cancel symmetrically. This case corresponds to a (private-key) cryptosystem that is used for both private-key encryption and message authentication. We do not know to reduce this case to the previous ones. Nevertheless, all our positive results extend also to protocols having such operators.

New Note: *All the additional cases are dealt with by reduction to the treatment of the forms studied in section 2.2 or by extension of that study. However, as shown in subsection 2.2.1, restricting the forms to some categories may yield a simpler treatment. Our intention, stated in a laconic note from 1985, was to study the power of operators that cancel asymmetrically.*[9] *Recall that we have shown that the set of form $\{E, D, a, d\}$ exhibit all complications that must be dealt with for a general set of forms, and that restricting the forms to $F_E \cup F_D$ makes life much simpler. Our plan (in 1985) was to study the effect of restricting the set of forms to forms that cancel symmetrically (i.e., $F_E \cup F_D \cup F_{fun} \cup F_{inv}$), but it seems that this plan was not carried out (or at least we currently fail to find any record of an actual study of this issue).*

2.4.2 The Insecurity Problem When Allowing Improper Instances

Throughout Chapter 2, we assumed that honest users refuse to take part in improper instances of protocols. Recall that a $\bar{u}$-instance of a ping-pong protocol is said to be improper if the sequence $\bar{u}$ contains two (or more) occurrences of the same user. The above assumption underlied the definition of insecurity (Definition 9 in subsection 2.1.3), where only proper instances of protocol words were considered. In this subsection, we omit this restriction and consider (for the insecurity definition) also improper instances.

Definition $9'$ **(weak-insecurity):** *Let $P[\bar{x}]$, $\bar{u}$ and Σ_J be as in Definition 9.*

- *For every $J \subseteq I$, let $\text{INST}'(P, J)$ denote the set of all instances of protocol words of P in which the users are from J; that is, $\text{INST}'(P, J) = \{\alpha_j[\bar{v}] : 1 \le j \le l, v_1, v_2, ..., v_p \in J\}$.*
- *Protocol $P[\bar{x}]$ is* weakly-insecure *if there exist a set $S \subseteq I \setminus U$, and an operator string $\gamma \in (\Sigma_S \cup \text{INST}'(P, S \cup U))^*$ such that $\gamma \alpha_1[\bar{u}] \equiv \lambda$.*

The only difference between Definition 9 and Definition $9'$ is that while $\text{INST}(\cdot, \cdot)$ (in Definition 9) contains only proper instances, $\text{INST}'(\cdot, \cdot)$ (in Definition $9'$) contains also improper instances. We believe that Definition 9 is a much more natural than Definition $9'$, nevertheless let us now consider the notion of weak-insecurity (according to Definition $9'$).

Proposition X1: *With respect to multi-party ping-pong protocols, insecurity implies weak-insecurity, but weak-insecurity does not imply insecurity.*

[9]The power of such operators was demonstrated in Section 6.2 of our technical report [EG], but our intention in 1985 was to investigate their affect on the number of saboteurs that should be considered in determining the insecurity of protocols.

Proof: The implication is trivial. For the non-implication, consider the following 2-party protocol: $((x_1, E_{x_2}a_{x_2}), (x_2, d_{x_1}D_{x_2}))$. This protocol is weakly-insecure (e.g., $d_{u_2}D_{u_2} \cdot E_{u_2}a_{u_2}$ is a weak-insecurity string demonstrating this fact). The reader can verify that the above protocol does not have an insecurity string and thus is not insecure. ■

Proposition X2: *If a multi-party ping-pong protocol is weakly-insecure then a single saboteur suffices to demonstrate it.*

Proof: Consider a S-weak-insecurity string of the protocol, denoted δ. Let s be an arbitrary user in S. Replace, in δ, all operators indexed by $r \in S$ by operators with the same form indexed by s. The resulting string is a $\{s\}$-weak-insecurity string of the protocol. ■

Thus, for every fixed p, the weak-insecurity of p-party ping-pong protocols can be tested in $O(n^3)$ time and $O(n^2)$ space (n is the length of the input), by using the techniques described in [DEK] (and sketched in the Appendix). However, testing the weak-insecurity of multi-party ping-pong protocols is NP-Hard (for details consult [Itz]).

2.4.3 Testing Insecurity of a Word in Presence of a Protocol

We consider the following generalization of Definition 9.

Definition 9* (insecurity of words in presence of a protocol): *Let $P[\bar{x}]$, Σ_J and* $\text{INST}(\cdot,\cdot)$ *be as in Definition 9. Let $\beta \in \Sigma^*$ be an operator word such that for infinitely many $m \in \{0,1\}^*$, $\beta(m)$ is defined. Let U be the set of indices of the operators of β. The work β is* insecure in presence of the protocol $P[\bar{x}]$ *if there exist a set $S \subseteq I \setminus U$ and an operator string $\gamma \in (\Sigma_S \cup \text{INST}(P, S \cup U))^*$ such that $\gamma \equiv \beta$.*

Definition 9 is a special case of Definition 9* (setting β to the left inverse of $\alpha_1[\bar{u}]$).[10] In general, $\beta = \beta_2\beta_1$, where all operators in β_1 have right inverses, and all operators in β_2 have left inverses (otherwise $\beta(m)$ may be undefined for all $m \in \{0,1\}^*$). Let β_1^{-R} (resp., β_2^{-L}) denote the string which results by concatenating the right (resp., left) inverses of the operators in β_1 (resp., β_2); the result is a right (resp., left) inverse of β_1 (resp. β_2). Clearly, an operator word γ is equivalent to β if and only if $\beta_2^{-L}\gamma\beta_1^{-R} \equiv \lambda$. Testing β-insecurity is thus reduced to testing whether there exists an appropriate γ such that $\beta_2^{-L}\gamma\beta_1^{-R}$ is equivalent to λ. This can be done using the methods presented in [DEK] (and sketched in the Appendix).

2.4.4 Finding the Shortest Insecurity String

Returning to Definition 9, one may ask:

> *What is the shortest insecurity string of $P[\bar{u}]$?*

or

> *does $P[\bar{u}]$ have an insecurity string of length smaller than q?*

[10]In fact, in case $\alpha_1[\bar{u}]$ does not have a left inverse, the protocol is secure in a trivial but usdeless way; that is, it cannot be used to transmit the initial message to any other party.

In this subsection we present efficient algorithms for solving the above questions.

> New Note: *The following discussion assumes that p and $|S|$ are constants.*

The length of the shortest S-insecurity string of $P[\bar{u}]$ can be found using a modified version of the Dolev-Even-Karp algorithm [DEK]. First (as in [DEK]) the automata accepting the regular language $\Delta(P, U, S)^* \alpha_1[\bar{u}]$ is constructed. A path in the automata is called collapsing if the string that corresponds to it is equivalent to λ. Next shortest collapsing pathes between pairs of states of the automata are iteratively found, using a *priority* queue.[11] The number of insert/delete-min operations on the queue is at most n^2 and the number of decrease-key operations is at most n^3, where n is the number of states in the automata. Implementing the priority queue by a Fibonacci Heap [FT] yields running time $O(n^3)$ (and space $O(n^2)$).

To find a shortest S-insecurity string itself, pointers should be left during the above algorithm. This will later allow to reconstruct the insecurity string in time linear in its length. Further details can be found in [EG].

> Old Note: *Length of insecurity string versus length of S-insecurity string???*

> New Note: *Indeed, we may set S_0 such that $|S_0| = 3p - 4$, but it may be the case that more saboteurs allow to obtain shorter insecurity strings. Definitely, the number of saboteurs in the shortest insecurity string is smaller than the length of the S_0-insecurity string, but out algorithm is exponential in the number of saboteurs...*

[11]In [DEK], arbitrary collapsing paths were iteratively found using an ordinary queue.

Chapter 3

Two-Party Extended Ping-Pong Protocols

3.1 Definition

New Note: *A memo dating to July 1998 reads: The half-word operators where defined as follows. For any* $\mathtt{op}$ *defined before (e.g.,* E, D, i, d*), we define* $\mathtt{op}^L, \mathtt{op}^R$ *to operate on strings over* $\{0, 1, \$\}$*, where* $\mathtt{op}^L$ *(resp.,* $\mathtt{op}^R$*) is undefined for strings in which the number of* \$*'s is not 1, and* $\mathtt{op}^L(w'\$w'') \stackrel{\text{def}}{=} \mathtt{op}(w')\$w''$ *(resp.,* $\mathtt{op}^R(w'\$w'') \stackrel{\text{def}}{=} w'\$\mathtt{op}(w'')$*), where* $w', w'' \in \{0, 1, \}^*$.

The paper also defined $\mathtt{op}^W$*, where* W *stands for* whole*, but I think this is not needed. (The definition is the natural extension of* $\mathtt{op}$ *to the new alphabet).*

New Note: *Security is defined as before. That is, the adversary is still restricted to attacks as in Section 2.1.3. In particular, it cannot mix pairs of strings (which represent messages) to obtain a new pairs (which was not obtained as such in prior steps). The latter restriction is probably justified by the use of the forms that operate on the entire word (specifically* E^W *and* D^W*), but this justification seems to rely on the use of these operators in the protocol. It seems that the specific protocols used in the reduction (below) satisfy this intuition. Anyhow, the bottomline is that even though the forms* $\mathtt{op}^W$ *are not required for the reduction, they are essential for justifying the definition of security of the protocols used in the reduction.*

3.2 Undecidability

New Note: *This is proven in [EG] by reduction from the Post Correspondence Problem. See pages 24–28 of [EG].*

Chapter 4

Conclusion

New Note: *This chapter was written in 1985. The original report [EG] contained nothing of this sort.*

When studying the security of cryptographic protocols, one can take one of the following two approaches:

1. Distinguish between the security of the "high level structure" of the protocol and the security of the cryptosystems used for its implementation. While studying the security of the (structure of the) protocol, it is assumed that the protocol is "implemented" with "ideal" cryptosystems (i.e. it is assumed that the cryptosystems are free of any properties which are not implied by the cancellation of encryption with the corresponding decryption).

   This approach can be found in [NS], [DY], [DLM], [DEK], [EG] and [EGL].

2. Study the security of a concrete implementation of the protocol with respect to the concrete cryptosystems used for the implementation.

   This approach was pursued in [LMR], [BGM] and [GMR].

   New Note: *My intention in 1985 was to strongly advocate the latter approach.*

References

[BGM] Ben-Or, M., Goldreich, O., Micali, S., and Rivest, R.L., "A Fair Protocol for Signing Contracts", to appear in the *proceedings of the 12th ICALP*, 1984.

[CR] Church, A., and Rosser, J.B., "Some Properties of Conversion", *Trans. Amer. Math. Soc. 39*, (1936), pp. 472-482.

[DLM] DeMillo, R., Lynch, N., and Merritt, M., "Cryptographic Protocols", *Proc. of the 14th ACM Symp. on Theory of Computation*, 1982, pp. 383-400.

[DH] Diffie, W., and Hellman, M.E., "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.

[DEK] Dolev, D., Even, S., and Karp, R.M., "On the Security of Ping-Pong Protocols", *Inform. and Control*, Vol. 55, 1982, pp. 57-68.

[DY] Dolev, D., and Yao, A.C., "On the Security of Public-Key Protocols", *IEEE Trans. on Inform. Theory*, Vol. IT-29, 1983, pp. 198-208.

[EG] Even, S., and Goldreich, O., "On the Security of Multi-Party Ping-Pong Protocols", TR No. 285, Computer Science Dept., Technion, Haifa 32000, Israel, June 1983, (59 pages).

[EGL] Even, S., Goldreich, O., and Lempel, A., "A Randomized Protocol for Signing Contracts", *Advances in Cryptology: Proceedings of Crypto82*, (Chaum D. et. al. eds.), Plenum Press, 1983, pp. 205-210. To appear in the *Comm. of the ACM.*

[FT] Fredman, M.L., and Tarjan, R.E., "Fibonacci Heaps and their uses in Improving Network Optimization Algorithms", *Proc. of the 25th IEEE Symp. on Foundation of Computer Science*, 1984, pp. 338-346.

[GJ] Garey and Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., 1979.

[GM] Goldwasser, S., and Micali, S., "Probabilistic Encryption", *Jour. Comp. and Sys, Sci.*, Vol. 28, 1984, pp. 270-299.

[GMR] Goldwasser, S., Micali, S., and Rackoff, C., "The Knowledge Complexity of Interactive Proof-Systems", *Proc. of the 17th ACM Symp. on Theory of Computation*, 1985, pp. 291-304.

[Itz] Itzhaik, Y., "A Protocol-Word Problem which is NP-Complete", private communication, 1983.

[LP] Lewis, and Papadimitriou, C.H., *Elements of the Theory of Computation*, Prentice-Hall, Inc., 1981.

[LMR] Luby, M., Micali, S., and Rackoff, C., "How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*, 1983, pp. 11-21.

[NS] Needham, R.M., and Schroeder, M.D., "Using Encryption for Authentication in Large Networks of Computers", *Comm. of the ACM*, Vol. 21, No. 12, 1978, pp. 993-999.

[PY] Papadimitriou, C.H., and Yannakakis, M., "The Complexity of Restricted Spanning Tree Problems", *Jour. of the ACM*, Vol. 29, April 1982, pp. 285-309.

[Po] Post, E.L., "A Variant of a Recursively Unsolvable Problem", *Bull. of the Amer. Math. Soc.*, 52, 1946, pp. 264-268.

[RSA] Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Comm. of the ACM*, Vol. 21, February 1978, pp. 120-126.

[Ro] Rosen, B.K., "Tree-Manipulation Systems and Church-Rosser Theorems", *Jour. of the ACM*, Vol. 20, No. 1, January 1973, pp. 160-187.

Appendix: The DEK Algorithm for Testing Insecurity

We review the Dolev-Even-Karp algorithm for testing insecurity of two-party protocols [DEK], and comment on its adaptation to the multi-party case.

> New Note: *This was the intension, but the current text merely defines the formal language problem to be solved. It then assumes that the reader knows how to solve this problem (see Way 1 below) or knows how this (formal language problem) is solved in [DEK] (see Way 2 below).*

By Definition 9 (Section 2.1) and our results of Section 2.2, for every fixed p, testing insecurity of p-party ping-pong protocols can be reduced to the following word problem: *does there exist and a string $\delta = \gamma\alpha_1[\bar{u}]$ in the set $\Delta(P, U, S)^*$ such that $\delta \equiv \lambda$, where S is an arbitrary subset of $I \setminus U$ having cardinality $3p - 4$.*

Once $P[\bar{u}]$ is given, the set $\Delta = \Delta(P, U, S)$ is finite and thus Δ^* is a regular expression over $\Sigma(F, U \cup S)$. Furthermore, by Section 2.2, $|\Delta| = O(p) + \frac{(4p-4)!}{(3p-4)!} \cdot l$, when l denotes the number of protocol words, and the total length of the words in Δ is $O(p) + \frac{(4p-4)!}{(3p-4)!} \cdot n$, where n is the length of the protocol. On the other hand, note that the set of words $\{\beta \in \Sigma(F, U \cup S)^* : \beta \equiv \lambda\}$ can be generated by a context-free grammer. Thus, the above word problem is reduced to *testing whether or not the intersection of a regular expression and a context-free language is empty.*

There are two ways one may proceed in solving this problem.

1. First construct a grammer for the intersection, and next test whether it generates a non-empty language. This algorithm requires time and space that are cubic in the total length of the words in Δ. For a constant p, we get running time (and space) that is cubic in the length of the protocol.

2. First construct an automata accepting the regular set, and next construct the "collapsing relation" induced by the grammer on the automata. For more details see [DEK]. Note that the automata has size linear in the total length of the words in Δ, and so (for constant p) the DEK algorithm will run in cubic time and linear space (in the length of the protocol).